Aron Insinga

Inspection Moderator's Workshop

Quality Development Program



digital™

+-------------+
d	i	g	i	t	a	l
+-------------+

I N T E R O F F I C E M E M O R A N D U M

Date: 9 July 1984
From: Lou Cohen
Dept: SAC Central Quality Group
Ext: DTN 381-1339
Loc/Mail: ZKO 1-2 / D13

INSPECTOR'S HANDBOOK

This handbook describes in detail the process of Formal Inspection as it is used at DEC. If you are about to participate in a Formal Inspection as an Inspector, you will find this handbook very helpful. If you intend to serve as a Moderator please first read this handbook and also attend the Inspection Moderator's Workshop, a course sponsered by the SAC Central Quality Group.

Many thanks to all the contributors who helped put this document together. Special thanks to Dana Chabot, Ron Randall, Hank Moran, Pam Saloky, Roger Allen, Mike Rasmussen, everyone in the SAC Central Quality Group, and almost everyone in the Methods and Tools Group.

This revised version of the Handbook incorporates material that has been covered in the Inspection Moderator's Workshop, but has never been put into written form before. Most of the new material relates to Process Control questions, that is questions about what to do with the data collected from inspections, besides just fixing the problems found.

In addition, the new inspection report forms have been included. These forms were redesigned when the software tool IDAT was completed. They make it easy for members of a project team to input inspection problem data into an IDAT database.

Finally, a glossary and a bibliography have been added at the end of the document.

1.0 A BRIEF LOOK AT FORMAL INSPECTIONS 4
1.1 What Are Formal Inspections? 4
1.2 Why Are Inspections Important? 6
1.3 What Are The Advantages Of Inspections? 7
1.4 Formal Inspections Versus Informal Walkthroughs . 7
1.5 A Look At Costs 8
1.6 Inspections At DEC 9
2.0 THE PHASES OF AN INSPECTION 10
2.1 Planning 10
2.1.1 Sizing The Target Document 12
2.1.2 Entrance Criteria For Inspections 14
2.2 Overview 15
2.3 Preparation 18
2.3.1 Categorizing Inspection Problems 18
2.3.2 VISIBILITY Problem Type 20
2.3.3 PRESENCE Problem Type 21
2.3.4 QUALITY DIMENSIONS Problem Type 21
2.3.5 ENGINEERING Problem Type 25
2.3.6 Nits 28
2.4 Inspection 29
2.4.1 Decision To Reinspect 30
2.5 Report Results 31
2.6 Rework 32
3.0 RESPONSIBILITIES OF PARTICIPANTS 33
3.1 RESPONSIBILITIES OF INSPECTORS 33
3.2 RESPONSIBILITIES OF MODERATORS 33
3.3 Hints On How To Run A Meeting 34
3.4 RESPONSIBILITIES OF THE READER 36
3.5 RESPONSIBILITIES OF THE RECORDER 38
3.6 How To Record A Problem 38
3.7 RESPONSIBILITIES OF THE AUTHOR 39
4.0 RESOURCES REQUIRED FOR AN INSPECTION 40
5.0 WHAT TO DO WITH INSPECTION PROBLEM DATA 41
5.1 Introduction To The Seven Basic Tools 41
5.2 Brief Descriptions Of The Seven Basic Tools . . 42
5.3 The Seven Basic Tools And Formal Inspections . . 48

APPENDIX A INSPECTION GUIDELINES - REQUIREMENTS

APPENDIX B INSPECTION GUIDELINES - DESIGN

APPENDIX C INSPECTION GUIDELINES - EASE OF USE

APPENDIX D INSPECTION GUIDELINES - GENERAL CODING

D.1 FUNCTION D-1
D.2 FORM D-1
D.3 ECONOMY D-2

APPENDIX E INSPECTION GUIDELINES - BLISS

E.1 COMPILE PROBLEMS E-1
E.2 RUN TIME PROBLEMS E-1

APPENDIX F GUIDELINES FOR TEST PLANS

APPENDIX G GUIDELINES FOR TEST CASES

APPENDIX H GUIDELINES FOR INSPECTION RATES

APPENDIX I SAMPLE PROBLEM REPORT FORM

APPENDIX J PROBLEM CATEGORY CODES AND OTHER CODES

APPENDIX K SAMPLE INSPECTION REPORT FORM

APPENDIX L GLOSSARY OF TERMS

APPENDIX M BIBLIOGRAPHY

1.0 A BRIEF LOOK AT FORMAL INSPECTIONS

1.1 What Are Formal Inspections?

Formal Inspections are a method for improving the quality of our products and increasing the productivity of our engineers.

The Formal Inspection is a type of technical peer review in which a small group of engineers, led by a trained "Moderator", carefully examines a project document, line by line, in order to find problems. The inspection process includes a method for both recording and categorizing problems. The purpose of recording problems is to give the author of the document the information needed to fix them. The purpose of categorizing them is to give the project team the opportunity of detecting generic types of problems or generic blind spots in order to prevent problems from occurring in the future. Here are some important points about inspections:

1. The supervisor of the author selects and assigns the inspectors. Each inspector should be able to contribute unique technical expertise to the inspection. The supervisor does not normally attend the inspection.

2. Typically 3-5 people attend a single inspection. Inspections last about 2 hours, during which 500 lines of text or 250 lines of code (excluding comments) can be inspected.

3. The inspection is codified and structured, in order to insure efficiency and uniformity.

4. In every inspection, the "target document" (the document being inspected) is compared with one or more "source documents" (the document(s) from which the target was derived).

   Additional "supporting documents" may also be present at an inspection. These are documents that should have had an influence on the work that produced the target document, but that may not have been the primary focus of attention.

   The Source Documents will have already been inspected. The following table gives some examples of Source and target documents:

SOURCE DOCUMENT	SUPPORTING DOCUMENTS	TARGET DOCUMENT
Product Requirements	Business Plan	Product Specification
Product Specification	Product Requirements	Development Plan
Psuedo-code for module	Product specification Product architecture document Project goals (from Product specification)	Code for module

(Inspection of the lead-off requirements document is a special case, discussed elsewhere).

5. A "problem" in the target document is any statement or the absence of any statement which makes the target document inconsistent with source documents or the supporting documents. In other words, the source and supporting documents are treated as the complete basis for the contents of the target. For the purposes of the inspection, they are assumed to be completely correct. In this way, we have an objective standard for what is right and what is wrong in the target document.

   In real life, of course, it often happens that errors occur in the source documents. In fact, these errors are often first discovered at inspections. The inspection process works best, however, when the source and supporting documents are as reliable as possible. For this reason, it is recommended that the source documents be inspected and reworked before the target document document is inspected. One of the hidden advantages of the inspection process is that it encourages project teams to put more than customary effort into the completeness and correctness of the early project documents. (There is a special problem, however, caused by the fact the first document in any project has no source document to compare it with, and so must be inspected in a special way. See the Appendix entitled "Inspection Guidelines - Requirments".)

6. As problems are discovered, they are recorded and classified by problem type (several categories). The list of problems belongs to the author; the problems by category are entered into a database for analysis. In a later section you will read about methods and tools you can use to make sense of the data available from inspections.

7. The responsibility for correcting problems, and for generating the creative solutions that may be required, is the author's.

8. Conducting many "reviews" at DEC is thought of as a method for provoking creativity and engineering innovation. Inspections are not intended to replace that function.

   Under inspections, the normal creative engineering process continues as it does today. But after all strategic decisions have been made at a particular engineering phase, we would expect to see the inspection process employed as a true Quality-Control function, the primary role of which is to help ensure that the designs and technical decisions that have been made are implemented without error.

9. Inspections occur AFTER controversy has been resolved. Inspections are not intended to help people resolve controversy; in that sense, inspections are not a substitute for poor management. Instead, inspections provide a way to help Engineers develop the product has been already decided upon.

1.2 Why Are Inspections Important?

There are several principles of Quality Control and Process Control that lead us to promote inspections.

1. Within any process, problems come from two sources: Common Causes (assignable to the system, that is, the total working environment) and Special Causes (assignable to individuals or special cases). Authorities on the subject estimate that 85 percent of all problems are due to Common Causes. Since we can begin to solve problems only when we understand their causes we must first distinguish between problems having Common Causes and problems having Special Causes.

   There is only one way to begin to understand which problems are due to Common causes: we must observe what kinds of problems occur at what points in the engineering process, and with what frequency. Then, by statistical methods, we can begin to identify and fix the problems that have Common Causes (the generic problems).

   Inspections give us an opportunity to perform such problem analysis.

2. W. Edward Deming states that we can minimize our testing costs by doing either no testing or 100 percent testing, depending on whether:

   (K2/K1)*P is less than 1 (no testing) or greater than 1 (100 percent testing) where:

   K1 = unit cost of testing,

   K2 = unit cost of letting a defect remain in the product, and

   P = average proportion of defects in a product at a particular stage of development.

   Today at DEC we do not have values for K1, K2, or P. The inspection process allows us to track costs and frequencies so that we can begin to estimate these critical variables.

3. Although we do not have exact values, we believe that it is cost effective to find and fix problems at the earliest possible point in the engineering process.

   Inspections give us a way of identifying problems early. We can therefore cut total project costs while increasing our chances of building high quality products on schedule.

1.3 What Are The Advantages Of Inspections?

1. Manageability. Inspections allow us accurately to identify when project milestones have been met, because when a project document passes inspection, we know that it is both complete and as problem-free as we can make it with our current technology.

2. Uniformity. The inspection Process is structured and codified. This tends to reduce the effects of personalities. We therefore have a uniform and objective method available for judging the effectiveness of our engineers. Eventually, we may acquire automated quality control tools to take the place of some inspections, but until then, inspections are our best bet.

3. Transferability. The inspection process can be easily taught to people. (However, inspection Moderators must be selected and trained).

   The inspection process can be introduced to engineering groups quickly. Successful inspections are not limited to our consulting engineers; they are available to everyone. As our engineering organizations expand, we need techniques that are easily transferable to engineers with all levels of experience, and that effectively increase quality and productivity. The Formal inspection is one such technique.

1.4 Formal Inspections Versus Informal Walkthroughs

There are many styles and types of reviews held at DEC. What follows is a list of attributes of inspections. As you read them, ask yourself: which of the reviews or walkthroughs that you have attended have had ALL of these attributes?

1. Leader (Moderator) has formal training for running the review.

2. Moderator "drives" all aspects of the meeting, from planning through to the optional re-review.

3. Each participant has a definite role during the meeting.

4. Checklists available to help participants identify problems.

5. Historical problem data available to guide participants.

6. Followup to check on rework is a formal part of the process.

7. Detailed feedback to author increases chances for fewer FUTURE problems.

8. Analysis of problem data used to influence process changes and to reduce generic problems.

1.5 A Look At Costs

Formal inspections are new to DEC, so data about costs is not plentiful. But there are early indications that at DEC inspections will easily pay for themselves. Here are some cost figures taken from a series of 6 inspections of a DEC software engineering project:

	Design/Spec	Debug/Test	Product Shipped	Project Totals
Defect Costs Of Project As A Result Of 6 Inspections:				
Number problems found and fixed	169	0	0	169
Unit Cost ($)				
..of finding problems	21.30	10.00	100.00	
..of fixing problems	3.31	30.00	1600.00	
Total Cost ($)				
..of finding problems	3600.00	0.00	0.00	3600.00
..of fixing problems	560.00	0.00	0.00	560.00
..of finding and fixing problems	4160.00	0.00	0.00	4160.00
Estimated Defect Costs For Same Project If It Had Not Had Those 6 Inspections:				
Same problems (found and fixed later in project)	0	165	4	169
Unit Cost ($)				
..of finding problems	21.30	10.00	100.00	
..of fixing problems	3.31	30.00	1600.00	
Total Cost ($)				
..of finding problems	0.00	1650.00	400.00	2050.00
..of fixing problems	0.00	4950.00	6400.00	11350.00
..of finding and fixing problems	0.00	6600.00	6800.00	13400.00

NOTES:

1. 169 problems were found by the 6 inspections.

2. In the opinion of the project leader, one of the problems found by inspection would definitely otherwise have been overlooked until the time of packaging for the SDC and would have delayed the project by 2-4 weeks.

3. In the opinion of the project leader, 4 of the problems found would definitely have never been noticed except by inspection, and would have resulted in unique-bug SPRs at an average cost to answer of $1600 (FY82 cost estimate).

4. Some of the other 165 problems found probably would have been found later on but prior to release and would have been fixed at an average cost higher than $3.31. This example assumes an average unit cost of $10 to find each such problem, and $30 to fix each such problem.

1.6 Inspections At DEC

A number of organizations in DEC have shown interest in inspections. Some major product development groups are using inspections for all new projects. Others are experimenting with inspections in a number of ways. The types of products involved are: Operating systems, Generic applications, System tools and utilities, Firmware, and LSI chips.

There is now enough interest in this methodology to justify a course which teaches engineers how to be inspection Moderators. This document serves as the "text book" for the inspection Moderator's Workshop.

2.0 THE PHASES OF AN INSPECTION

Each inspection has 6 phases:

1. PLANNING
2. OVERVIEW
3. PREPARATION
4. INSPECTION
5. REPORT RESULTS
6. REWORK

2.1 Planning

1. The Author initiates the process by deciding that his or her work is ready for inspection. On many projects, the "doneness" criteria for deliverables is: document complete, document inspected, reinspection decision is for no reinspection, and the rework has been completed. Therefore, the Author allows time in his or her schedule for all of this to happen, and sets the expected completion date accordingly.

   The Author then advises the Supervisor or Project Leader that it is time for the inspection. At that point, responsibility for the inspection passes from the Author to the Supervisor. The Supervisor then contacts an available inspection Moderator and a 15 minute meeting called the Planning Meeting occurs. (See below: "Entrance Criteria for Inspections".)

   The Planning Meeting involves the Supervisor, the Moderator, and the Author. Jointly, they determine:

   - Whether or not the target document is ready for inspection (see "Entrance Criteria for Inspections").
   - Regarding the target document:

     1. What is it?
     2. How big is it? How many 2-hour sessions will be required to inspect it? (See "Sizing the target document" below.)
     3. What is the Source Document?

4. What supporting documents will be required?

- The goals of the inspection (if other than the standard goal of finding problems of translation between the target document and the source document).
- Who will attend the overview and inspection.

Here are some guidelines for determining the best inspection team.

1. Try to keep the total number of participants, including the Moderator, between 3 and 5 people. The high-level documents often require 6 or 7 people, but the more people present, the more difficult it is to have a useful inspection.

2. Each person should bring a UNIQUE expertise to the inspection.

   Example: for an inspection of a Disk Driver routine, invite:

   The author

   The Moderator

   The Designer of the Disk or the Disk controller

   Someone who has written a similar routine

   Example: for an inspection of a Requirements document, invite:

   The author (presumably a Product Manager)

   The Moderator

   The Project Leader

   The lead documenter

   A representative from the Customer Support Organization (CSSE)

   The Program/Quality Manager

3. No one should be present at the inspection who is a supervisor of someone else in the room. The presence of a supervisor is bound to bias the results of the inspection. No one is immune from the concerns of trying to look good in front of the boss, or of trying to protect or show

someone up to the boss.

The only time this rule may be broken is when the supervisor has some unique expertise which NO ONE else can provide.

- Who will prepare the material for the overview.
- Who will present the material at the overview.
- When the overview will take place.
- How the "Reinspection Decision" will be made (see section: "Decision to Reinspect").
- Who will receive copies of the final report.

At the end of the planning meeting, responsibility for the inspection passes from the Supervisor to the Moderator, whose job it now is to ensure that all the remaining steps in the process proceed correctly.

2. Moderator prepares for overview:

   - Gets conference rooms for Overview and inspection, and informs attendees.
   - Prepares copies of Problem Categorization Sheets.

2.1.1 Sizing The Target Document -

The rate at which material is inspected is critical. Studies have shown that if the rate is too fast, costly problems will be overlooked. On the other hand, if the rate is too slow, money and resources will be wasted, and inspections will be so tedious that people will lose their motivation to participate. The Moderator must take the time to understand exactly how big the target document is.

By knowing exactly how many lines of text are involved the Moderator can estimate how much time will be needed for the entire inspection. If the inspection is likely to take more than two hours, the moderator must confer with the Author to divide the document into smaller pieces, each of which can be inspected in two hours or less. Dividing the document into smaller pieces sometimes takes a lot of ingenuity and creativity, since each piece must be self-contained enough so that it is comprehensible to the inspection team. If there is no way to divide the document into smaller inspectable units, then the document may not be inspectable, and one should ask whether anyone else could understand it in its present form.

Another reason why the Moderator should know how big is the document is to judge, during the inspection, whether the team is moving too slowly to complete the inspection within two hours. The two hour limitation is explained in the guidelines below. The moderator can also determine if the team is moving too quickly. If so, important problems in the document may be missed.

The best guidelines we have so far for effective use of inspector's time at an inspection is:

- Inspections should last no longer than 2 hours. This is because inspections require a great deal of concentration, and most people can sustain a high level of concentration for only about 2 hours. There are, to be sure, many who can maintain a high level of concentration for much longer periods of time, but not everyone, and not always. In an inspection we are asking several people to expend the same amount of effort for exactly the same period of time, and experience has shown that 2 hours of concentration is a practical expectation for a group of people.

  Incidentally, for similar reasons, there is a particular time of day which actually works better for groups than other times. The best time for an inspection is from 10AM until noon. Most people, whether they are owls or larks, can get to a 10AM meeting on time, and function reasonably alertly at that time. Naturally, not all inspections can be held at that time, but if there is a choice, try to schedule your inspection for 10AM.

- For code - any language - inspect at the rate of 125 NCSS per hour.

  NCSS= "Non-Commented Source Statements". These are lines of text within a compiled listing which are not entirely comments.

  There has been some confusion about what constitues a "line". Regardless of any programming language conventions or rules, a "line" for the purposes of inspection is the amount of text that appears on a line of paper (usually not more than 132 characters of text).

  The overriding concern for inspections is how understandable the text is to humans, not machines. Therefore we judge volumes of code according to human perceptions - lines of text.

- For any text which is not code to be processed by some translator, inspect at the rate of 250 lines per hour.

How do you decide if a target document has more or less lines than can be inspected in an hour? You MUST count them!!!! It is astonishing how inaccurate the estimates of line count are when people "eyeball" the documents. Experience has show that you can't go by page count, or average number of lines per page, or anything except a physical count of lines.

The following method seems to work well. (Programmable calculators are useful for this process).

1. Count the number of lines on each page and record that number at the bottom of the page. Real boilerplate, such as the table of contents, or the distribution list, if these are routine and not controversial, can be skipped. (If they ARE controversial, or if they will be inspected, they should be counted.)

2. Total the lines.

3. Check the Appendix "Guidelines for Inspection Rates". If the total number of lines in the document is much larger than 250 lines (for code), or much larger than 500 lines (for text), replan the inspection -- break it into multiple inspections, and ask the author for advice on how to subdivide it.

   If the volume of text is inspectable in a 2 hour period, go on to the next step.

4. Using the appropriate inspection rate of 125 or 250 lines per hour, estimate the time to inspect each page of the document and record the cumulative time (starting at T=0) at the bottom of each page.

Sounds like Mickey Mouse? Yes, and No. Experience has shown that this series of steps helps ensure that the target document is short enough to be inspectable, and that the Moderator can keep the inspection moving at exactly the right pace.

2.1.2 Entrance Criteria For Inspections -

Since inspections cost money, it is important to schedule an inspection at the right time.

If an inspection occurs too early, the target document may be changing substantially while it is being inspected, and the effort to inspect will be wasted.

If an inspection occurs too late, new documents may have already been produced based on the uninspected Source Document. The result may be that costly problems in the uninspected Source document will be propagated into new documents.

Therefore it is useful to have some guidelines (Entrance Criteria) to help decide on the most effective time for an inspection. The Moderator must use these guidelines with care. Often, but not always, an inspection held at the wrong time will still be more valuable than no inspection at all. Therefore some judgement is called for. The Moderator must consider the cost trades-off, the overall quality of the product, time-to-market issues, local politics and morale, to name the most obvious factors that enter into deciding on the right time to do an inspection.

Here are some guidelines to consider:

1. The Source Document must have been inspected before the target document can be inspected. If the Source Document has not been inspected, it will be extremely difficult for the inspectors to decide what is right and what is wrong in the target document. In fact, the inspections performed so far seem to indicate that the single greatest waste of project resources can be traced to faulty (uninspected) requirements and specifications documents.

2. To the greatest extent possible, any controversy surrounding the document must have been resolved. Inspections normally cannot resolve controversy. Instead, inspections are designed to find problems after strategic decisions have been made.

   On some projects, controversy is unavoidable. The Moderator must decide whether the inspection will help to settle any outstanding controversy (usually it won't), or whether the controversy will still exist after the inspection.

3. In the case of text, the author (and no one else) must state that the document is ready for inspection. It is never a good idea to impose an inspection on someone who is not ready for it.

4. In the case of code, we do not yet know what the best time is to schedule an inspection. There are 2 schools of thought:

   - The code should be as fully debugged as the programmer can get it. This is the time most programmers prefer to have their code inspected, and it is the time when most code at DEC gets reviewed.

     The other school of thought...

   - The code has been compiled correctly but has never been run on a machine. This is the strategy favored by many experienced inspectors. They argue that desk-checking, done by individuals prior to inspection, and done by a small group during inspection, is far more cost effective than "trial and error" debugging.

   We suggest that you try both methods, observe the results, and let us know what you think. When we have more data on the subject, we will be able to argue for one approach over the other.

2.2 Overview

1. The purpose of the Overview is to hand out materials, decide on formal roles, and walk through what is to be inspected (what it is and how it fits into the scheme of things).

The Overview is a kind of formal announcement to the inspectors that there is to be an inspection and that they have a job to do.

2. The Overview usually lasts 30 minutes or less. When all attendees are already familiar with the inspection process, and have been involved with prior inspections of the same project, the Overview can usually be completed in 10 minutes.

   Despite its brevity, the Overview should not be skipped. It helps to make the inspection itself run smoothly. Experience has shown that when Overviews are omitted, something usually goes wrong at the inspection. Typical problems that Overviews head off are:

   - Inspectors arrive at the inspection with the wrong materials.
   - Inspectors forget to prepare for the inspection.
   - Inspectors forget when the inspection was to be.
   - and many other "dumb" but expensive problems.

3. Attendees at the Overview are the inspectors (including the Moderator and author). Inspectors from distant sites or from out of town are normally excused from attending the Overview. Of course, the Moderator is responsible for ensuring that the out-of-towners have received the materials in plenty of time to do their preparation.

4. Sometimes the target document will be a "High Interest" document, where a large group of people would benefit from a presentation of the document's contents.

   In such a case, the Overview can be extended to be an "Information Only" meeting for a large group, which happens to include the inspectors. In this case, the duration of the Overview would be long enough to cover the material at a level of detail appropriate to the group.

   The Overview should be structured to be "Information Only". It is not the forum for finding problems, nor is it an open invitation for brainstorming. If brainstorming is still needed, the target document is probably not ready for inspection.

   The Overview IS a good time for attendees to ask questions that will aid them in understanding the material better. But the Moderator must watch out for rhetorical questions, which are in fact comments about the material. Example of such a rhetorical question: "Why did you use design technique X?", where the tone of voice suggests that anyone who uses design technique X must be a complete idiot. If there is a valid concern about the design technique, then there is still controversy, and the target document is not ready for inspection.

Also, the Moderator must make a judgement that the material is in fact understandable. If the questions (and the author's answers) seem to indicate that the material raises more questions than it answers, it may be appropriate for the Moderator either to request that the author rework the document or for the Moderator to replan the inspection before going ahead.

5. Activities at the overview:

   - o Moderator announces the Target, Source, and supporting documents.

   - o Moderator announces the date(s), times(s), and location(s) of the inspection(s).

   - o The author describes in general terms what is contained in each of the documents. The author should limit the description to the organization and conventions used in the documents, and to the way in which the documents relate to each other. The author should plan to talk for not more than 10 minutes. (Exception: if the Target is a "high interest" document, as described above).

   - o Other action items at the Overview:

     - The author hands out all materials to each inspector. (No last minute changes allowed after the Overview!)

     - The author announces the order in which the material will be inspected.

     - The author gives hints on how to read the material.

     - The Moderator selects a Reader and a Recorder. (Best method: ask for volunteers).

     - The Moderator hands out the Problem Categorization Code sheet (see Appendix: "PROBLEM CATEGORIES AND TYPES") and explains it to anyone who is unfamiliar with it. This can be done after the "inspection veterans" have left the room.

     - The Moderator explains the inspection process to anyone who is unfamiliar with it. This can also be done after the "inspection veterans" have left the room.

     - The Moderator confirms that each person has the time to prepare for the inspection (1 to 2 hours for each 2 hours of inspection).

     - The Moderator stresses the need for punctuality at the inspection.

2.3 Preparation

Inspectors should spend 1 to 2 hours in advance of the inspection reviewing the target document, noting and categorizing problems. A definition for a "problem" is given above in the section entitled: "What are Formal Inspections?". See also the section below on "Categorizing Inspection Problems".

The Moderator should check with the inspectors at least 2 hours ahead of the inspection to see if they will be prepared. The Moderator must decide whether enough preparation has been done to cost-justify the inspection.

If the preparation has not been adequate, the best thing to do is to postpone the inspection and to report the reason to Management. The inspection Form has space reserved for this possibility.

2.3.1 Categorizing Inspection Problems -

The person who finds a problem is responsible for categorizing it. This section deals with the problem categories used for inspections.

Eventually, the specific problem categories will change. This is because the inspection process will help us to understand better what types of problems actually occur, and what types of problems inspections are good at finding.

If you have need to modify the problem list for your development group, please notify the Central Quality Group. In this way, the CQG can keep an updated list of error codes which will be made available to everyone. Also, CQG intends to publish error data analysis in order to help all of focus on the generic problems in our engineering process. This can only be done meaningfully when we all use a common set of codes. If you invent new codes, please be sure that they are unique 2-letter codes.

Each problem should be categorized 4 different ways. The four Types of Problem categories are called Visibility, Presence, Quality Dimensions, and Engineering types.

The Types are generally independent of each other, but don't let it bother you if occasionally the Types seem to overlap.

Similarly, the Categories within a Type are intended to be mutually exclusive and all inclusive, but the selection of terms used is not perfect and it never will be.

Each problem type has been assigned a unique 2-letter code. The codes have been chosen for their mnemonic value where possible, but some of the "mnemonics" may stretch your imagination a bit. When assigning codes to a problem, try to select the one code from each of the four Problem Types which best describes the problem. Exception: for the Quality Dimensions, use up to 2 codes (see below).

If more than one code within a problem type seems to apply, try to pick the code which applies the most.

Experience has shown that the Quality Dimensions overlap considerably; that is, a single error often affects more than one Quality Dimension. Therefore, it is helpful to specify up to TWO Quality Dimension codes when categorizing a problem. The second Quality Dimension Code is optional.

At the moment, we don't think we need to to allow for more than one code for each of the other Problem Types, although we will expand our record-keeping mechanism if the need is there.

The notation for problems would look like this:

"Missing check for legal unit number VI/MS/RE/FE/LO"

Explanation: The VISIBILITY code is "Visible" (VI) because the problem would be visible to a user who had specified an illegal unit number;

the PRESENCE category is "Missing" (MS) because there is no check for this case;

the first QUALITY DIMENSION Category code is "Reliability" (RE), because the program would probably crash if an illegal unit number had been specified;

the second QUALITY DIMENSION Category code is "Functionality" (FE) because the missing check affects the way the program functions as well as its reliability;

and the ENGINEERING CATEGORY is "Logic" (LO) because there is missing logic in the program to take care of the case of an illegal unit number.

It is convenient to record all the problems on a special form (see Appendix: "Sample Problem Report Form".) During an inspection this form would be filled out manually by the Recorder, although the example following has been typed in order to ease the problems of reproduction of this document.

For more on how to use this form, see the section: "How to record a problem".

Problem ID	Problem Description	VI/IV	MS/WR/EX	Qual	Qual	Eng	Multiples	Discoverer
1	Missing check for legal unit number.	VI	MS	RE	FE	LO		JO
2	Misspelled command name.	VI	WR	PB		EP	//	LC
3	File is always searched from beginning, even though next record needed will always occur beyond current accessed record.	VI	WR	PE		LO		

2.3.2 VISIBILITY Problem Type -

1. Code VI (Visible to a user; previously called MAJOR). A VI problem is a problem in the document being inspected that would, if uncorrected, cause the program to malfunction, or cause someone using the document to make an error that would cause the program to malfunction.

   The malfunction could be trivial, such as a misspelled word in an error message, or catastrophic, such as a system crash following login. The important distinction for this category is that the END USER would see the problem if it were not corrected. Many people are initially confused by the VI and IV codes. This is because we tend to think that the user of a design document is the person who implements the design. The purpose of the VI and IV codes is to help us remember the customer. The Engineering type codes, described below, give us a way to focus on the problems from the engineer's popint of view.

   It is more meaningful, however, to think about problems at every phase of a project in terms of how they would affect the end user if they were not fixed. In this way, our consciousness of the customer will be enhanced. Also, by categorizing and then counting problems found according to how they affect the end user, we can better estimate the contribution of the inspection process to the bottom line: ultimate product quality.

   Many internal engineering costs are not well understood at Digital, but we usually do know the cost to Digital of problems found by our users. By classifying problems found as visable or not visable, and by counting the visable ones and multiplying the count by the average cost saved due to a problem prevented, we can begin to assess the cost-effectiveness of the inspection process.

2. Code IV (Invisible to user; previously called MINOR). IN problems are problems that are important, but that the USER probably would not see.

   Examples are maintainability problems, internal standards violations, stylistic inconsistancies, and incorrect comments.

2.3.3 PRESENCE Problem Type -

1. Code MS (MISSING) problems are omissions in the target document. MISSING problems include missing statements about how a product should behave under certain conditions. Also, missing logic, cases or program states not accounted for, missing functions, and so on.

2. Code WR (PRESENT, BUT WRONG) problems are incorrect statements in the material being inspected. This category has in the past been called simply "WRONG", but some people object to the tone and implications of the word. If you prefer, please feel free to call the category "PRESENT". (Old-timers will probably still call it "WRONG".)

   Example: a Product Specification document (Target) states that the operating system should support up to 128 terminals when in fact what was asked for in Product Requirements (Source) was support for 256 terminals. The number of terminals to be supported was not missing, but the number was wrong. (VI/WR/EN/FE/GO)

3. Code EX (EXTRA) problems are statements in the target document that are superfluous; if omitted, no harm would be done.

   Example: The Specification (Target) states that the Fortran compiler should support a set of language constructions for real-time processing, but the Requirements Document (Source) says nothing about it. (VI/EX/FE/CO/GO)

2.3.4 QUALITY DIMENSIONS Problem Type -

1. Code CO (Costs) relates to a problem which affects the budget for the current project, or the price of the product.

   Example: the Development Plan (Target) calls for 5 Software Engineers, but the budget (Source) only allows for 3 Software Engineers. (IV/WR/CO/GO)

   Another example: the Specification (Target) states that the software product requires 3 RL02 disk drives, but common knowledge (or a perhaps a recent marketing study (Source)) indicates that 90

percent of our customers have fewer than 2 RL02 drives. (VI/WR/CO/GO)

2. Code CM (Compatibility) relates to a problem in terms of the compatibility of this product with other DEC products and data processed by other DEC products, command language, conventions, and so on. Topics covered under this heading are: compatibility at the conceptual, user interface, program interface and base operating system interface levels.

   "Compatibility" covers compatiblity with other DEC products, with previous and subsequent versions of this product, with DEC standard user interfaces, with other DEC device handlers, with other DEC utilities, and with DEC applications programs.

   It also covers standards conformance, for the company, industry and for international standards.

   Example: the Specification (Target) states that three existing command formats will change in the new release, but the Requirements document (Source) states nothing about compatibility with the previous release. Since, in most cases, upward compatibility is a requirement unless a specific requirement for non-compatibility has been stated, this is called a Compatibility problem. (VI/WR/CM/FE/GO)

3. Code CN (Constraints and trades-off) relates to a problem which affects the product with respect to any special constraints or priorities. "Constraint" refers to some action which MUST be done without which there would be no viable product. "Trade-off" refers to a decison to achieve some goal at the expense of some other goal. Note that "contraints" can never be traded-off.

   Example: the Specification (Target) states that three existing command formats will change in the new release, but the Requirements document (Source) states that complete compatiblity with the previous release is a CONSTRAINT of the project. (VI/WR/CN/CM/GO)

4. Code EN (Environment) relates to a problem with respect to the environment in which the product will be used. The "environment" includes the Users, the Hardware, the Software and the Services.

   Example: the Requirements document (Target) failed to describe the range of hardware configurations this product should run on. (VI/MS/EN/FE/GO)

5. Code EV (Evolvability; extensibility) relates to a problem which effects the ability of this product to be eventually adapted to suit changing customer environments; or, a problem which affects the future enhancements, or migration paths of the product.

   Example: the Design (Target) states that this utility program will build an in-memory table of data derived from the input file. This is an Evolvability error because although the input files are

known to be small, there is no guarantee that the input files will continue to stay small. Therefore the in-memory table may overflow as customer's input files grow. (VI/WR/EV/FE/EP)

6. Code FE (Software capabilities; features; functionality) relates to a problem which would affect anything that can be observed from the outside of the product by a person, by hardware, or by other software. It includes all inputs and outputs such as commands, qualifiers, switches, parameters, files, screen layouts, and messages. It also includes less obvious items such as sense switches, lights, peripheral modes, voltages, device positioning, calling sequences and condition codes. It also includes formatting and validation rules.

   Example: a Specification (Target) for a set of Screen management routines fails to provide a way for the user to clear the screen. (VI/MS/FE/GO)

7. Code HF (Human factors; ease of use) relates to a problem which would affect characteristics that tailor the human interface of the product to fit the needs of the target user. Factors often considered include the behavior of the software and the content and organization of the publications.

   Example: A Specification Document (Target) defines seven commands for a utility program. The first 4 letters of each of the seven commands is identical. (VI/WR/HF/EP)

8. Code IN (Installability) relates to a problem which would affect the users perception of the product from the time the package is opened, to the time the product can be put to the use it was purchased for.

   Example: the Specification (Target) does not address product Installation. (VI/MS/IN/GO)

9. Code MN (Maintainence (answering SPRs)) relates to a problem which would affect the SPR answering capability of the organization which will be responsible for SPRs and maintenance. Most questions of Maintainence are management questions.

   The Requirements Document (Source) states that all priority 1 SPRs will be answered within 7 days of receipt by Engineering.

   The Development Plan (Target) indicates that SPRs will be answered by the newest member of the team, and no time has been allotted to train her, nor does any maintainence documentation exist. (VI/MS/MN/GO)

10. Code MT (Maintainability) relates to a problem which would affect ease of diagnosis and correction of problems in the software and the publications.

Example: the labels defined in a module of code (Target) have no mnemonic value, and no comments exist to describe them. (IV/WR/MT/PC)

11. Code PE (Performance) relates to a problem which would affect the characteristics the product will display in specific environments and under specific conditions. Areas usually considered are: workloads, speed of operation, bandwith, performance trades-off, efficiency, and expected usage of main and auxiliary memory.

    Example: a Design Document (Target) states that a certain file is always searched from the beginning, even though it can be shown that successive records required can always be found after the last record accessed. (VI/WR/PE/LO)

12. Code PK (Packaging) relates to a problem which would affect initial presentation and appearance of the package, to a user. Issues usually included under this heading are: software distribution media, publications format, and appearance of the packaging container.

    Example: the Development plan (Target) allows no time in the schedule to create and certify the master product kit. (VI/MS/PK/EP)

13. Code PB (Publications) relates to a problem which would affect the purpose, content or organization of any book in the publication set.

    Example: A Specification Document misspells a command name. The author has not misspelled this command name elsewhere, but the Specification Document (Target) is to be the primary source document for the Software Documentation Engineer, who is writing the User's Manual. (VI/WR/PB/EP)

14. Code RE (Reliability) relates to a problem which would affect the user's perception of the dependability and predictability of the product. Areas of product performance which come under this category are: level of errors, number of patches required, on-line availability, error handling, response to system failure, product safety and corporate liability, SPR rate, and unique defect discovery rate.

    Example: the Specification (Target) has no discussion of how the product should behave when the user's disk quota has been exceeded. (VI/MS/RE/EP)

15. Code TM (Timeliness) relates to a problem which affects the schedule constraints of the project.

    Example: despite a Requirement (Source) that the product must ship on January 1, the schedule in the Development Plan (target) shows the ship date as March 1. (VI/WR/TM/EP)

16. Code HS (Not classified, can't decide)

For more detailed information on these Quality categories, consult the guidelines in the Engineering Plan Policy, Software Development Policies and Procedures number 7A8-4.

2.3.5 ENGINEERING Problem Type -

The ENGINEERING type codes relate to the engineering process itself. By analyzing the incidence of various engineering problems over the course of a project, a project team can begin identifying which engineering problems occur many times, and are therefore targets for elimination by changes in the engineering process. Because different project teams have different priorities and different problems, these category codes will not apply equally to all teams. The list here should be considered a sample, heavily biased toward software development. Project teams will probably want to start with the present list and then modify it as their experience with inspections increases.

1. Code AM (Statement ambiguous) relates to a statement in the target document which could be interpreted more than one way and could therefore cause a problem on the project or to the end user.

   Example: "The button is located on the side of the cabinet." The statement does not specify which side of the cabinet, and could be misinterpreted by both engineers, writers of user documents, and users. (VI/WR/HF/AM)

2. Code DI (Data initialization) relates to the failure of the target document to specify the initialization of data used elsewhere, correctly.

   Example: a variable in a program must be initialized to zero, but it is not initialized at all. (VI/MS/RE/FE/DI)

3. Code DD (Data description or data structure) relates to a problem in the definition of data or data structures used by the product.

   Example: the Design Document (Target) defines a field as 7 bits, but the same document implies that the field must take on 255 values. (VI/WR/RE/FE/DD)

4. Code DU (Data usage) relates to a problem in the use of data or data structures.

   Example: the module (Target) accesses a field of 30 bits, but it was defined in the design document (Source) to be 26 bits. (VI/WR/RE/FE/DU)

5. Code EP (Explanation or definition) relates to a statement or statements which attempt to describe or explain a process or concept.

Example 1: The Specification Document (Target) contains no mention of performance characteristics of the product, although the Requirements document (Source) states that the product should perform competitively with IBM product X. (VI/MS/PE/EP).

Example 2: The Design document (Target) contains a rationale for a certain algorithm, and because the rationale is mixed in with the actual design material, the readers of the Design Document are confused. (IV/EX/FE/MT/EP)

Example 3: The Design Document (target) contains reference's to Swinton's Parsing Algorithm, but neither defines it nor provides references to its definition. (VI/MS/RE/FE/EP)

6. Code GH (Goal in higher level requirements) relates to a problem in a supporting document which was discovered during this inspection.

   Example: The following two statements appear in the Requirements (Source):

   "This compiler will conform to ANSI spec XXXX."

   "This compiler will accept all IBM COBOL programs without modification."

   Since this particular IBM COBOL product contains non-ANSI features, the two statements are in conflict. The conflict between the two statements is not noticed until the inspection of the Specification Document (Target). (VI/WR/FE/GH)

7. Code GO (Goal) relates to an inconsistency or other problem in the target document with respect to explicit or implied requirements in the Source or supporting documents.

   Example: The Requirements Document (Source) states: "This compiler will conform to ANSI spec XXXX.", while the Specifications Document (Target) states: "This compiler will accept all IBM COBOL programs without modification." (VI/WR/FE/GO)

8. Code IC (Inconsistant statement) relates to two or more statements in the target document which are in conflict with each other. Neither statement by itself is a problem, but the conflict between them creates a problem.

   Example: The Requirements Document (Target) states: "This compiler will conform to ANSI spec XXXX." Elsewhere in the same document is the statement: "This compiler will accept all IBM COBOL programs without modification." Since this particular IBM COBOL product contains non-ANSI features, the two statements are in conflict. (VI/WR/FE/IC)

9. Code LI (Loop initialization).

Example: A DO...UNTIL loop is used, but DO...WHILE should have been used. (VI/WR/RE/FE/LI)

10. Code LO (Logic) relates to a logic problem.

    Example 1: "IF statement" goes the wrong way. (VI/WR/RE/FE/LO)

    Example 2: A segment of code in a module (Target) may be interrupted during execution, but the code assumes that a field, which is available to other processes, remains unchanged during execution of the module. (VI/WR/RE/LO)

11. Code MX (Module interdependency) relates to any problems which occur when one module interacts with another.

    Example: Module A (Target) calls Module B (supporting document) with incorrect parameters. (VI/WR/RE/MX)

12. Code NS (Not specific enough) relates to material in the Target document that is too general or too vague.

    Example: The Design Document (Target) contains the statement: "The XYZ routine will provide a reasonable interface to all routines which call it". The use of the word "reasonable" leaves too many questions unanswered for a Design Document. (IV/WR/RE/MT/NS)

13. Code OO (Off by one).

    Example: The design calls for addressing table entries as offsets from entry 1, but the base entry should have been entry 0. (VI/WR/RE/OO)

14. Code PC (Programming convention) relates to a violation of applicable programming standards.

    Example: Module name (Target) is defined as AXLGRS, but programming standards require that all modules of this type have a "$" as the first letter of their names. (IV/WR/MT/PC)

15. Code PV (Passive voice) relates to a statement made ambiguous because it was written in the passive voice.

    Example: "If the AST is deliverable and the process is the current process, the ASTLVL register is also updated." The reader cannot determine how the check for AST being deliverable is made, or when it is made, or what module is responsible for making the check. Similar unanswered questions exist for actions regarding the ASTLVL register. (VI/WR/RE/FE/PV)

16. Code SY (Syntax) relates to a statement in the target document that is ambiguous because it is not a correct English sentence.

Example: "Also the ASTLVL register." The reader cannot tell what kind of reference is being made to the ASTLVL register because it is not clear what is the subject, verb or object of the sentence. (IV/WR/MT/RE/SY)

17. Code TS (Too specific) relates to a level of detail innappropriate for the target document.

    Example: The Requirements Document (Target) names routines to be altered, people to do the job, and schedules. Such a statements focus on the solution to a problem, but Requirements Documents should be stating what is needed without dealing with how the needs are to be met. (VI/WR/HS/TS)

18. Code UE (Use of English) relates to a statement in the target document that is not clear and is not categorized by codes PV or SY.

19. Code OT (Other)

2.3.6 Nits -

Often people spend costly time pointing out obvious errors, which, if corrected, would not contribute to the success of the project. Time is wasted in three different ways:

1. Time is wasted when the "problem" is identified. Often, when a nit is presented as if it were problem, people will feel the necessity to debate whether it's a problem or not.

2. Time is wasted when the "problem" is recorded.

3. Time is wasted when the "problem" is fixed.

Another harmful byproduct of calling out a nit is that people sometimes lose sight of the real problems when they are forced to look at nits.

For example, a product specification contains a typographical error: The word "and" is misspelled as "nad". Taken in context, it is clear to everyone that the word should have been "and". This is a NIT. Nothing is to be gained by mentioning the problem at an inspection.

Note: the same misspelling in a User Manual or in the text for an error message is NOT a nit. There is the possibility that some user will notice the misspelling and create an SPR because of it. In that case, the trivial misspelling could cost DEC hundreds of dollars, because the average cost to DEC engineering for SPRs, even duplicates, is $1000.

In inspections, avoid calling out NITs!! They waste everybody's time.

Caution: it is not always obvious whether or not a problem is a NIT. When in doubt, call it out. But when it's clearly a NIT, put your energy into finding the real problems. One development team came up with a novel way of dealing with nits, because everyone on this team was especially fond of finding and discussing them. They created a special version of the problem report form in which several "generic" descriptions of common nits were predefined and printed on the form. In this way, whenever anyone called out a nit at an inspection, the error description and categories could simply be selected from the existing preprinted ones. Since it was not possible to suppress the nits, the project team found a rapid method for handling them.

2.4 Inspection

1. The inspection should begin and end promptly on schedule. Contrary to conventions at many other DEC meetings, inspections start exactly on time and never run overtime.

2. Seating: the Moderator sits at the head of the table; the Reader sits near the Moderator. The author sits at the opposite end of the room. This arrangement is often helpful to shy authors, because the Reader, sitting on the opposite side of the room, will focus attention away from the author.

3. The Reader reads or parpahrases the target document aloud to the group, line by line.

4. Anyone who sees a problem or has a question interrupts the Reader when the appropriate section of the target document has been read.

   It is not necessary to arrive at concensus about whether something is truly a problem. Extended discussion can often be avoided by recording problems even if there is disagreement, and then moving on. This is not to say that discussion should always be avoided; sometimes discussion can lead to better understanding of the target document and discovery of more problems. But it's up to the Moderator to decide whether a discussion is likely to lead to better problem identification, or whether it is pointless. If it's pointless, and someone insists there is a problem, the Moderator should ask the inspector to state what the problem is and to classify it. Then move on.

   The person who identified the problem should also categorize it. There may be discussion at this point as well.

5. As problems are identified, the Recorder lists them, along with the problem categories.

6. The Moderator checks the time against the time-notations s/he has made on each page of the target document. The Moderator encourages the group to slow down or speed up as necessary. The Moderator must help the group avoid unproductive discussion. S/he

must encourage the group to identify "problems, not solutions".

7. After all problems have been identified and recorded, but before the inspection is adjourned, a decision is made regarding reinspection.

2.4.1 Decision To Reinspect -

This fundamental decision answers the question: "After the rework to correct the problems found has been done, is it necessary to inspect the rework?". From management's point of view, if reinspection is not required, the target document may be regarded as complete. If reinspection is required, the target document may be regarded as not yet complete. The factors which affect this decision are:

1. The complexity of the target document
2. The complexity of the expected rework
3. The abilities of the author

Unfortunately, these are difficult factors to weigh. Some objective criteria or some simple method is needed to make the decision to reinspect. Here is a list of possible methods for deciding on reinspection. The Central Quality Group is in favor of method 1, but we are interested in any experience anyone has with any of the methods.

Methods for deciding on reinspection:

1. The author decides. This is consistent with the DEC management philosophy which gives each engineer as much responsibility and authority as possible. The Central Quality Group is in favor of this method.

   Recent experience has shown that most Authors rarely or never decide to have their material reinspected. Some Authors, when asked about this later, admit that a reinspection probably would have produced a better end result. The Central Quality Group invites your opinions on this subject.

2. If any single inspector wants reinspection, the target document must be reinspected after rework. The rationale here is that since each inspector knows how important his/her decision is, he/she will make a careful, responsible decision. Since each inspector was selected because he/she had some unique expertise, his/her decision carries some unique authority and must be adhered to.

3. Entire inspection team decides by consenus. This is the ideal, and in practise most inspection teams will have little difficulty coming to concensus. Unfortunately, there will always be a few times when concensus is not possible. In such cases, some other

method for deciding on reinspection is needed.

4. If the number of problems found exceeds 5 percent of the number of lines in the target document, the target document should be reinspected after rework. This method is used at IBM, where they claim that only 5 percent of their documents require reinspection. It would seem to be a very unlikely method to appeal to DEC engineers, but it apparently gets results at IBM.

   Suggestion: be sure you have settled on a method for deciding on reinspection BEFORE the inspection starts. See the section above on Planning.

2.5 Report Results

1. At the end of the inspection, the Recorder photocopies the inspection Problem Report (sample form in Appendix at end of this document) and gives one copy to the author, and one to the Moderator.

2. At the end of the inspection the Moderator completes the inspection Report (sample form in Appendix at end of this document). One copy should be hand-delivered to the author, another to the Supervisor, and a third should be sent, along with the inspection Problem Report, to the Central Quality Group, MLO3-5/B39.

3. Note that the inspection Problem Report does NOT go to the Supervisor or to any else on the project. The nature and number of problems found during the inspection is nobody's business but the people attending the inspection. In fact, confidentiality is an important aspect of the inspection Process. The process is designed to help people find and fix problems in a product design as cheaply and effectively as possible. It is NOT designed to give management an opportunity to measure people based on the number of problems found in their work. If it were used that way, people would be motivated NOT to find the problems!

   The author needs the detailed problem information in order to fix the problems and to do any analysis of problem types s/he would like to.

   The Central Quality Group needs the data to merge it with inspection data coming from many sources, and to report to the company at large about GENERIC trends which the data suggests.

   The Supervisor needs to know only whether or not the inspection took place, how much it cost, and whether a reinspection is needed. That is the kind of management information Supervisors are accustomed to getting now, although the accuracy of such information will be much improved by the inspection process.

If a reinspection has been called for, the Moderator is responsible for scheduling it and for going through all the steps of an inspection again. If there is to be a reinspection, use the same inspectors and inspect only the rework.

2.6 Rework

The Author is responsible for reworking the document after the inspection. In Digital tradition, the Author is free to resolve problems identified at the inspection any way he or she sees fit. Digital engineers are assumed to be responsible individuals who will deal with the problems identified on their merits. Thus, the inspection cannot and should not be used by an inspector as a club to force the Author to act in a certain way. Should there be a disagreement about some issue in a target document, some sort of resolution should be arrived at before the inspection begins. AS has been stated earlier, the inspection process is not designed to resolve controversy.

3.0 RESPONSIBILITIES OF PARTICIPANTS

3.1 RESPONSIBILITIES OF INSPECTORS

1. Prepare for the inspection. Spend at least one hour, but not more than two hours, in becoming thoroughly familiar with the materials to be inspected. As a guide to how much time should be allowed for preparation, see the Appendix: "Guidelines for Inspection Rates".

2. Before reading the target document, read the appropriate "Inspection Guidelines", in the Appendices to this document. These guidelines are designed to remind you of important things to look for when preparing for an inspection. They should not be used rigidly. Some project teams may wish to modify or replace them for their own purposes.

3. Make a list of all problems you have found during your preparation period. Next to each problem noted, make a note of the problem categories.

4. Look for problems, NOT solutions. As soon as you say: "This SHOULD be such-and-such", you are proposing a solution. Proposed solutions are obviously helpful and important, but they should be discussed outside of the actual inspection. The most efficient use of time during your preparation period and during the inspection is in identifying, but not correcting, problems.

5. Get to the inspection meeting ON TIME. The notion of "on time" includes being ready to work (coffee cup filled, papers in hand, body in seat, pencil in hand.) The most important advantage of inspections is that they yield productivity gains. Don't offset the gains by making others wait for you.

6. During the inspection, mention problems and categories as you see them. If there is doubt or disagreement, the best solution is to state clearly what is wrong (NOT how it should be fixed), assign the error category codes, and move on.

7. When making comments at the inspection, remember that it is easy to offend other people. Please try to keep your comments to the job at hand: finding problems. Stay away from criticizing people.

3.2 RESPONSIBILITIES OF MODERATORS

1. Plan for the inspection (see the section above: "Planning").

2. Schedule and conduct the overview (see the section above: "Overview"). Invite the participants. Ensure that everyone has all the materials, checklists, and other information needed to do the job.

3. Prepare for and attend the inspection as would an inspector.

4. Run the inspection. Keep people on the track: "problems, not solutions". Avoid nits. Identify the real problems. Be on guard against offensive language or hurt feelings.

5. The Moderator has a special responsibility to the author. Experience has shown that no author is immune from irritation and defensiveness during an inspection. The problems fly at him or her relentlessly. To offset the inevitable reaction, the Moderator must find every opportunity to help the author feel at ease. Well placed humor, sympathy, and a sense of fairness all help.

   Each Moderator will have to find the ways, consistent with his/her own personality, to make the inspection a positive experience for everyone involved, so that the group works together, inspection after inspection, for the benefit of the project.

6. Get a decision on reinspection before the meeting ends. Get an estimate from the Author for completion of the rework.

7. Distribute results of the inspection (the Inspection Report), within 24 hours of the inspection. In most cases the reports should be distributed within 10 minutes of the inspection.

8. Follow up on rework; it must be done on schedule.

9. In general, do whatever is necessary to ensure the efficient use of everyone's time, and the quality of the target document.

3.3 Hints On How To Run A Meeting

1. Make sure that the inspectors have prepared for the inspection. At least 80 percent of all unsuccessful inspections can be traced back to lack of advance preparation.

2. As the inspection progresses, make certain that each inspector has an opportunity to comment on the product. People who have no chance to contribute at an inspection are not likely to volunteer for future inspections. Furthermore, the "quiet ones" often have things to say that no one else thought of.

3. Make sure that the inspection is properly paced. The Guidelines For Inspection Rates are designed to help you do this.

4. Learn to identify the "rat holes" and the "nits", and keep the group focussed on the real problems.

5. Regarding "rat holes":

   - If the inspection has gone off-track, try to discover whether someone is simply trying to conceal lack of preparation. If lack of preparation is a significant problem, it may be best to cancel the inspection, report the cancellation and the reason to management (there's room for this on the Inspection Report), and reschedule the inspection.

   - If the discussion is becoming overly detailed, acknowledge the point and ask the Recorder to make a note of it. This usually helps people to focus on the problem, and the recording of it acts as a natural ending point.

   - If that doesn't work, try reminding the group that the goal is to find "Problems, not solutions", since many "rat holes" are being dug toward pet solutions. It may help to request that the solution be given to the author off-line.

   - NEVER PUT ANYONE DOWN. (You'll lose more inspectors that way...) Always assume that each person, no matter how obstreperous, is trying to make a contribution. such an assumption sets a constructive tone, which most of the other inspectors will pick up on.

   - Watch out for long explanations. A good principle to observe is that each target document, together with its Source and supporting documents, should be reasonably self-contained. If long explanations are necessary, then either someone hasn't prepared, or the material is not sufficiently self-contained. As Moderator, you must determine which is the case, and decide on the most constructive course of action.

6. How to deal with people who are truly obstructive (from "The Ethnotechnical Review Handbook", by Weinberg and Freedman).

   "...when you start out with reviews, you may find people who are determined to sabotage the whole process by making a mockery of early reviews. This isn't a pleasant situation, but there's no need for the review leader to be intimidated by it.

   "Once you have convinced yourself that someone is really TRYING to be destructive, you should simply announce that this review is no longer being productive and is therefore terminated. The leader ALWAYS has the prerogative to terminate the meeting, so there can be no argument with this approach. Having terminated the meeting, however, you will be obliged to report to management the REASON why a good review wasn't obtained. Therefore, you will have to be prepared to defend your conclusion that a certain person was acting obstructively. This acts as a check on arbitrary judgement by you, the leader.

"If this obligation of the leader is carried out fairly and conscientiously in the earliest reviews, your installation is unlikely to experience this kind of difficulty in later reviews. The obstructors will either learn how to be constructive participants, or else stop being invited to reviews - or perhaps stop being invited to work at all. On the other hand, if a particular review leader habitually accuses other people of obstructing reviews which are actually being badly led, management cannot fail to notice before long. There is nothing to keep any review member from making a personal report to management on the poor conduct of a review.

"We must emphasize that such obstructive tactics are VERY rare in a reasonably healthy organization, and almost entirely confined to early reviews, when leaders are inexperienced and a few recalcitrants may still be left over from the "good ol' days". In fact, an experienced leader, faced with an apparent obstruction, will first assume that the person is actually acting out some situation OUTSIDE the review situation itself. Perhaps a child is critically ill at home, or a spouse has just taken a lover. Perhaps the person has a terrible stomachache, or just come from an argument with the boss, or both. The remedy is to TRY reasonable steps to bring the meeting on track. Then, if reasonable efforts fail, to suggest that the meeting be terminated if it continues in the same vein. If that mild threat doesn't work, then it really isn't a problem for the review leader, but a problem for management, and the meeting should be terminated."

3.4 RESPONSIBILITIES OF THE READER

1. Prepare for and attend the inspection as an inspector would (see RESPONSIBILITIES OF THE INSPECTOR).

2. At the inspection, read aloud or paraphrase aloud the target document. Read everything in the target document except "boilerplate" (table of contents, cover sheet, index).

3. Reading ordinary text is not too difficult, but reading software, or logic diagrams, or other very technical material, takes special care. The Reader's job is to add meaning to what is already there. While the Author may be more familiar than anyone else with the target document, it is better to have a more unbiased reader. The Author can easily project his or her own blindspots onto the entire group.

   Therefore, even though it takes special effort, it is best to have the Reader be someone other than the Author.

   What constitutes "good" reading? Good reading should provide some interpretation and clarity to the technical material. It should allow the other inspectors to test their interpretation of the

material against that of the Reader.

Here's an example of both good and bad reading. Judge for yourself which reading style is more likely to allow the inspection team to find problems.

Example to illustrate code reading:
==========

TARGET DOCUMENT: DEFECTS.BAS (used as classroom example in Inspection Moderator's Workshop)

```
100 'DEFECTS/BAS          by Lou Cohen 1982
200 'A(,) AND B(,) ARE ARRAYS OF AMPLIFICATION MODELS
300 'A(I,1)=DEFECTS CREATED THIS PHASE
400 'A(I,2)=DEFECTS PASSED ON FROM PREVIOUS PHASE
500 'A(I,3)=DEFECTS AS A RESULT OF AMPLIFICATION IN THIS PHASE
600 'A(I,4)=EFFICIENCY PERCENTAGE OF DEFECT FILTER THIS PHASE
700 'A(I,5)=DEFECTS IN PRODUCT AT END OF THIS PHASE
800 'AL(), BL() = RELATIVE RESOURCES EXPENDED IN THIS PHASE
900 '
1000 DIM A(5,5),B(5,5),AL(5),BL(5)
1100 NA$(1)="Requirements"
1200 NA$(2)="Spec and Design"
1300 NA$(3)="Coding until programmer ready to test"
1400 NA$(4)="All testing and debuging"
1500 NA$(5)="Field Test"
1600 N1$="% of testing time which is constant overhead"
1700 N2$="Ratio: Cost of fixing errors during testing, over Cost of fix
     ing errors any time before testing"
1800 N3$="Percent of total resources to be devoted to inspections"
1900 '
.
.
.
```

One method of reading (not recommended):

"Line 100. Apostrophe DEFECTS slash BAS by Lou Cohen 1982. Line 200. Apostrophe A left parenthesis comma right parenthesis and B left parenthesis comma right parenthesis are arrays of amplification models. Line 300. Apostrophe A left parenthesis I comma one right parenthesis equals defects created this phase. Line 400"....[etc.]

Another method of reading (much better):

"Line 100 is a comment giving the program name, DEFECTS slash BAS, the author's name and what seems to be the year in which the program was written. Line 200 is another comment. This line states that two 2 dimensional arrays, called A and B, are to be amplification models. Lines 300 to 700 are comments which describe the types of information to be stored in the Ith row of

array A, elements A sub I comma 1 through A sub I comma 5"....[etc.]

3.5 RESPONSIBILITIES OF THE RECORDER

1. Prepare for and attend the inspection as an inspector would (see RESPONSIBILITIES OF THE INSPECTOR).
2. At the inspection, use the "Problem Listing" form (sample in Appendix) to record all problems found.
3. As each problem is identified, use the technique described below to record it.
4. Give the original Problem Listing to the author.
5. At the end of the inspection, make a copy of the Problem Listing and send it to the Moderator.

3.6 How To Record A Problem

The following method seems to work well. It is easy to do, and it captures all the important information. You may want to glance at the sample form in the Appendix: "Sample Problem Listing Form" and the example of its use in the section: "Categorizing Inspection Problems" as you read this section.

1. As each problem is identified, assign it a number and write down the number in the "Problem ID" column, the problem description in as many lines as needed in the "Problem Description" column, and the 4 or 5 problem codes in the columns reserved for them on the Problem Listing form.
2. Enter the same number on a blank copy of the target document at the point where the comment about the problem applies. Make sure the number stands out, so the author can easily find it. Some people draw a circle around the number to make it more visible.
3. Many times the same problem will occur repeatedly during an inspection. For instance, a subroutine call may appear 15 times, always with the parameters in the same wrong order. Very few people have the inclination to record the same problem over and over again, and yet the author should be told explicitly of each occurrence, so that s/he will not overlook any of the problems during rework.

To deal with this case, simply record the number corresponding to the problem on the target document each place where the problem occurs, and put a check or a tick in the "Multiples" column for each additional time the problem is found. If the problem occurs only once, the "Multiples" column will be empty, if the problem occurs twice, there should be one tick mark, and so forth.

4. The left-most column of the Inspection Problem Report Form is reserved for the initials of the person who discovered the problem. Many times during rework the Author needs further clarification of the problem, and the initials help him find the person who understands the problem best.

3.7 RESPONSIBILITIES OF THE AUTHOR

1. Prepare for and attend the inspection as an inspector would (see RESPONSIBILITIES OF THE INSPECTOR).

2. Be on hand to clarify any issues that are raised.

3. Try to avoid defending your work. Rejoice in the fact that problems are being found now, and not later when project pressures will be much more extreme.

4. Make sure you understand every comment made.

5. After the inspection, resolve all the problems as quickly as possible. Keep in close contact with the Moderator.

4.0 RESOURCES REQUIRED FOR AN INSPECTION

The decision to implement inspections for a project must be made at a management level high enough to commit the resources necessary. The resources required are:

	Plan	Over-view	Pre-pare	Inspect results	Report problems	Fix	TOTALS
Manager	.5						.5
Moderator	1.0	.5	1.0	2.0	.5		5.0
Author	.5	.5	.5	2.0		X	3.5 + X
Inspector 1		.5	1.5	2.0			4.0
Inspector 2		.5	1.5	2.0			4.0
Inspector 3		.5	1.5	2.0			4.0
TOTALS	2.0	2.5	6.0	10.0	.5	X	21.0 + X

NOTES:

* All times are estimates in person-hours.

* Inspectors 2 and 3 are optional. Therefore, the total time for an inspection will vary from 13 person-hours to 21 person-hours depending on the number of inspectors.

* The value X stands for the author's time needed to fix problems found during the inspection.

  This time is dependent on the number and type of problems found, and should be charged to the development of the project, not to the inspection. In fact, the larger the value for X, the greater is the value of the inspection, because each problem found and fixed at this point in the project saves large amounts of time and money later on.

A Moderator probably spends about 4-1/2 hours on every 2-hour inspection.

Each inspector probably spends 4 hours on every 2-hour inspection.

Resources required for 100 percent inspection are estimated to be 15 to 20 percent of total project resources.

5.0 WHAT TO DO WITH INSPECTION PROBLEM DATA

5.1 Introduction To The Seven Basic Tools

In 1950, W. Edwards Deming gave a series of lectures to Japan's industrial leaders. His main emphasis was on product quality. He told the Japanese that a fundamental change was needed in their production methods: they must learn how to apply statistical tools to controlling the quality of their products. He also told them that the key to greater productivity was increased quality, and that only by understanding the production PROCESS was it possible to make improvements.

The Japanese took Deming seriously. The Japanese Union of Scientists and Engineers (JUSE) carefully selected a set of "statistical" problem solving tools and taught these tools to millions of workers. Over time, they evolved a list of "Seven Basic Tools" which had several unique properties:

1. They were easy to learn
2. They were easy to use
3. They provided a common problem solving language between company departments, and between companies (very important for support of the purchasing function)
4. They were especially effective in helping people understand their processes
5. They provided a basis for effective team problem solving

Because the Seven Basic Tools are so connected to team problem solving, the author has started to call them the "Teamwork Tools". Sometimes this term will be used in this text, in place of "Seven Basic Tools".

The Seven Basic Tools are:

1. Histograms
2. Cause and Effect diagrams
3. Check sheets
4. Pareto Diagrams
5. Graphs
6. Scatter diagrams
7. Control charts

What follows is a brief description of the tools. A word of caution, however: this is not a text book on the tools. It takes perhaps 8 to 10 days of classroom study to gain a basic working knowledge of all of the

tools. For detailed information, please read Dr. Kaoru Ishikawa's book: "Guide to Quality Control".

Certain tools are described here in some detail; others are barely mentioned. The more lengthy descriptions are here to support the examples of the application of certain tools to the formal inspection process. The tools recommended for use, and therefore described in more detail, are: Check Sheets, Pareto Diagrams, Cause and Effect diagrams, and U-charts. The remaining tools may also be useful, but at this point not enough is understood for us to recommend them. We do suggest that if you are interested in the examples given here, you take the trouble to learn about all of the seven tools, since they are reputed to work well in various combinations.

5.2 Brief Descriptions Of The Seven Basic Tools

1. Histogram. A histogram is a graphical representation of a frequency distribution by means of rectangles whose widths represent the class intervals and whose heights represent the corresponding frequencies.

   You can better understand this terse dictionary definition by looking at EXAMPLE 5-1. In the example, the class intervals are the numbers along the bottom row of the diagram. Each number stands for a class of inspections all having inspection rates (expressed in NCSS per hour) "nearest" that number. (You will recall from earlier parts of this handbook that NCSS stands for "non commented source statements". The number 110, for instance, stands for a group of code inspections in which the inspection rate for each was equal to or greater than 100, and less than 120 NCSS per hour. The number 110 represents a class interval, and each of the other numbers represents a different class interval.

   The columns of X's represent the rectangles. All the class intervals are equal in this example, so the widths of the rectangles are equal. The heights of the rectangles are represented by the number of X's, which in turn represent the average number of "major" errors found by the inspection teams that belong to that class interval.

   The example shows that for the data collected by F.O.Buck at IBM, the average number of errors found by inspection teams rose as the inspection rate rose from class interval 70 (60 to almost 80 NCSS per hour: 17 X's, or an average of 17 major errors found) to class interval 110 (100 to almost 120 NCSS per hour: an average of 21 major errors found). As the inspection rate increased beyond class interval 120 NCSS per hour, the average number of major errors found by inspection teams decreased. (Buck's study suggests that the inspection rate is one of the major factors affecting efficiency of inspection teams).

EXAMPLE 5-1

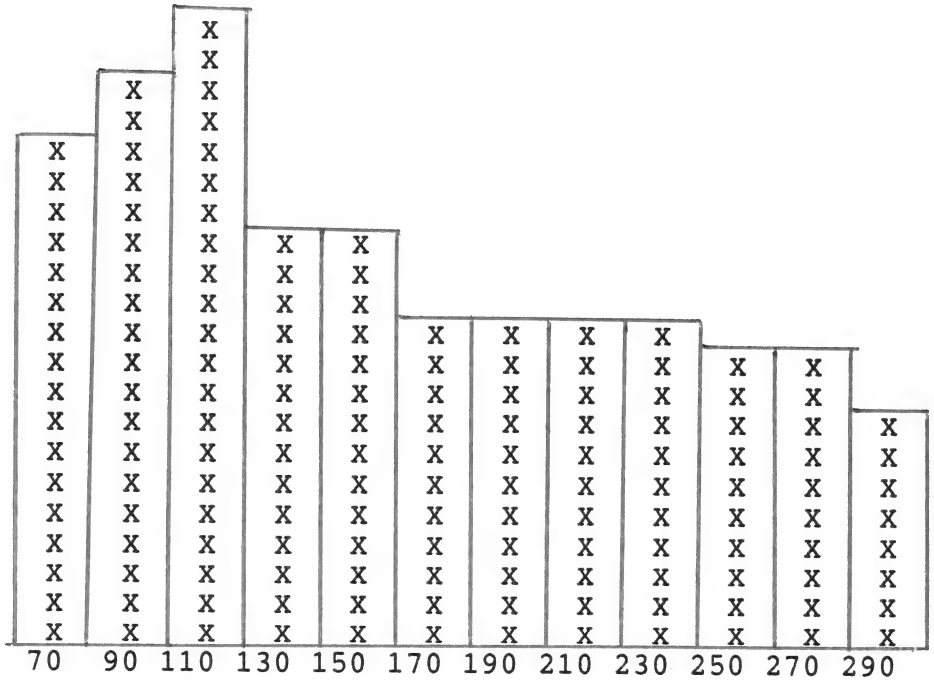
Average number of major errors found (X's)

versus

inspection rate in NCSS per hour

Each "X" represents one error found.

(based on IBM report TR 21.802,
"Indicators of Quality Inspections", by F.O. Buck.)



2. Cause and Effect diagram. This is a charting technique which helps a person or group to understand all the possible causes of a given effect. It is sometimes called an Ishikawa diagram or chart, because its inventer was Dr. Kaoru Ishikawa. It is also called a fishbone chart because it resembles a fish's skeleton. See example 5-4 below.

   In a Cause and Effect diagram, a main effect or phenomenon is represented by a horizontal line through the center of the diagram. The diagram reflects the idea that every effect is caused by one or more causes, and that each of the causes can be thought of as an effect in itself. For example, if the effect is: "Software products are expensive to buy", then one possible cause is: "Software products are expensive to produce". This cause can be thought of as an effect, which in turn has several causes, one of which might be "Labor costs are high". This process of

analyzing effects in terms of their causes is considered by the Japanese an essential step in fixing problems.

We draw a Cause and Effect diagram by representing causes for the main effect by diagonal lines coming off the central horizontal line. We then think of each cause as an effect, and we represent causes for that effect by lines coming off the diagonal lines. We extend the process by as many levels as we need in order to fully understand the process behind the original effect.

When the Cause and Effect diagram is used as a brainstorming tool (and that is one of its main purposes), the team members should be free to suggest any possible cause for an effect, without any filtering by other team members. The first goal is to identify and structure all possible causes. Later on, the team can select the most likely causes from the full list.

In the section below: "The Seven Basic Tools And Formal Inspections", the use of Cause and Effect diagrams is explained further.

3. Check sheet. This refers to two different tools: the first is a memory guide - a list of items to be checked off before a task or a process may be said to be complete.

   The second is a sheet for recording events that have occurred during some process. The events could be of many different types. Examples: number of employees at a plant by town; number of engineers with Masters degrees by department; number of shipments by product. Often in process control applications the events recorded are related to problems, defects or errors by some classification scheme.

   Diagram 5-2 is an example of a check sheet derived from data from a hypothetical code inspection. The example is explained in the section following, entitled: "The Seven Basic Tools and Formal Inspections".

4. Pareto Diagram. This is a graphical method for displaying the incidence of defects arranged in order of greatest importance. The number of occurrences (the left vertical axis), and the percentage of the total errors (the right axis) are displayed for each error category. The X's in the example below show number of defects. Each "o" shows cumulative percentage of all errors to the left of the column containing the "o".

   The Pareto Diagram gives a quick view of the most frequently observed error categories. Even though the most frequently observed category may not be the most serious category on a particular project, it may be wise to pay attention to it. This is because it is usually much easier to reduce the largest category of errors by half than it is to reduce the smallest category to zero. On the other hand, sometimes the severity of certain very low frequency problems is so great that they must be dealt with first. Common sense and business objectives must be

used to balance the costs and benefits of choosing one type or another of problem to reduce first, but the Pareto Diagram at least gives us a place to start.

Example 5-3 is an example of a Pareto Diagram. It was created from the hypothetical data shown below in the check sheet in Example 5-2.

5. Graph. This is any graphical technique for displaying data. Beside the Histogram, Pareto Diagram, Check Sheet, and control chart, each of which is a type of graph, many other graphs can be used for displaying data.

   Some other types of graphs are: broken line graph (points connected by line segments), curved line graph (points connected by curved lines), bar graph (use of rectangular "bars" to denote quantities), circle graph (circles divided into pie-shaped segments to indicate relative proportions), and combinations of these types of graphs to compare various relationships.

   For most purposes, the graphical display of data is easier to understand than are other ways of displaying data (such as long rows of numbers). For this reason, the list of the seven basic tools includes graphs specifically, so that we are less likely to ignore this powerful tool.

   For more on graphs, see Ishikawa, "Guide to Quality Control", chapter 6.

6. Scatter diagram. This is a graphical technique for showing the relationship between two measured quantities. For more information, see Ishikawa, "Guide to Quality Control".

7. Control chart. This is a graphical technique for displaying the measurable results of a process, and for determining when a measurement is "substantially" different from the main body of measurements.

   A control chart usually shows data arranged in chronological order. Each control chart contains three horizontal lines, called the "Central Line" (CL), the "Upper Control Limit" (UCL), and the "Lower Control Limit" (LCL). Each of these lines represents a numerical value, and is computed from a relatively small number of the earliest points plotted on the chart (just how many points to use is beyond the scope of this introductory description).

   These early points are representative of the "system", or the "set of constant causes" at work that are responsible for the variability between the points. If the causes remain constant, we can assume that the variability in the points will continue as it has been. The variability will change only if the causes change.

   Once enough points have been plotted, we compute the CL, UCL and LCL, and draw their values as horizontal lines on the chart. The CL is the average of the points plotted, and will therefore appear

in the midst of the points plotted. The UCL and LCL represent values above and below the CL, respectively, and are at a distance of three "standard deviations" from the CL. A theorem of statistics tells us that as long as the causes of variability in the plotted points remain constant, we can expect that 99.7 percent of all points will lie between the UCL and LCL.

This means that if we extend the UCL and LCL into the future on the control chart, and plot new points on the chart, then we should expect that nearly all new points will lie between these values. If a new point does NOT lie between the UCL and the LCL, then the chances are very good that the causes for variability in our process have changed. In other words, we have statistical evidence that the process that produced the new point is "substantially" different from the process that produced all the other points.

Walter Shewhart in 1931 stated that when a point falls outside of the UCL and LCL, the reasons for this extreme variability can usually be explained and eliminated. Shewhart called these reasons, or causes, "Assignable causes". In more recent years, Shewhart's term "Assignable causes" has given way to Deming's and Juran's term "Special causes". We shall use the term "Special causes" in this document. When we have eliminated all special causes, so that all new points fall within the UCL and LCL, we call the result a "process under statistical control".

W. Edwards Deming went further than Shewhart. He recognized that the variability in a process under statistical control may be predictable but nevertheless objectionable. When a process is affected by a set of constant causes, and yields unacceptable results, then the principle constant causes of variation must be understood and removed. If they are not, the results will remain unacceptable indefinitely.

Deming called the constant causes of variation "Common Causes". He made two important observations about the common causes for variation:

1. Only after removing special causes for variation is it possible to identify and remove common causes for variation. Among the reasons for this assertion is that the existence of special causes distorts the computation of CL, UCL and LCL.

2. The responsibility for removal of common causes for variation cannot rest with individual workers team members, since no individual worker can control causes that affect everyone. Therefore the responsibility for removing common causes for variation is the responsibility of the entire team, or of the team's manager, or of higher levels of management.

   A common error made by management is to attempt to solve problems of excessive variation without understanding the causes. Often management will try to eliminate variation that is really due to common causes AS IF the variation had been

caused by individuals (as if it were due to special causes). This often results in individuals being blamed for problems they cannot solve. The consequences are: the problem doesn't get solved, and one or a few workers are made more unhappy than they need to be.

Many different types of control charts have been devised by statisticians. Each has special applicability, depending on the type of data being collected. Data falls into two major classes: value data and attribute data.

Value data is measurable data that exists on a continuous scale. Examples of value data are: temperature, weight, length, time, voltage.

Attribute data is derived from observations of discrete entities or events. It is either the straightforward counts of these entities, or the rates of occurences of the entities, or the fractions of total occurences of the entities. Examples are: number of errors found per formal inspection (straightforward count); number of errors found per formal inspection, divided by the number of lines of text inspected (rate of occurence; see example of U-chart, Diagrams 5-5 and 5-6); fraction of Visible problems as compared to total problems found by inspection (fractions of total occurences).

Diagram 5-5 shows the computation of a U-chart (one type of control chart), and Diagram 5-6 shows the U-chart itself. You will notice that in Diagram 5-6 the UCL and LCL are not straight horizontal lines. This is because the data comes from several inspections. Since the sizes of the inspected documents vary one from another, the standard deviation varies somewhat with each inspection. The larger the document inspected, the smaller the standard deviation. A few of the other types of attribute control charts also have moving UCLs and LCLs. The principles of statistical control, common and special causes, still hold, however.

The Seven Basic Tools have stood the test of time. They are used on the production line, in the office, in sales situations, in engineering environments - in fact, their application in Japan seems to be extremely broad. While not all Japanese companies make full use of the tools, most Japanese companies we are familiar with, do. Such companies as Fujitsu, Hitachi, Toyota, NEC, and Sony are "believers", and promote the use of the tools at all levels within the company. The Japanese name for such widespread use of the tools is "Total Quality Control", or TQC.

Yokogawa Hewlett-Packard in Japan recently won the Deming Award. This means that they have completed a 5-year program aimed at TQC, and were judged to have been more successfull than other competing Japanese firms.

The Seven Basic Tools are used in the United States also. In response to the quality challenge of the Japanese, many companies, including General Motors, Ford Motor Company, Western Electric, Hewlett Packard and IBM have recently embarked on programs to teach large numbers of their employees how to use these tools. Western Electric plans to send 50,000 employees to a 5-day course in the use of the tools.

At Digital, a number of organizations, mostly in Manufacturing, are promoting the Seven Basic Tools. The tools are relatively new to Digital engineers, but a few groups are beginning to try to find ways of using them.

The formal inspection process provides an excellent basis for project teams to use the Seven Basic Tools, and to begin to approach the quality achievements of other companies that are using them.

5.3 The Seven Basic Tools And Formal Inspections

Four of the Seven Tools are quite easily applicable to the inspection process: Check Sheets, Pareto Diagrams, Cause and Effect diagrams, and U-charts (a species of Control Chart). Others of the Seven Basic Tools may also be applicable, but the author has not had enough experience to determine which ones and in which ways.

The methods for use suggested here should not be thought of as the "official" or only ways of using the tools. These are just suggestions. Like any good tools, their use is limited only by your imagination.

1. Check Sheets.

   You have already read that in the formal inspection process every problem found in an inspection is categorized several ways. These problem categories are recorded on the Inspection Problem Report Form. From these problem categories, you can create one or more check sheets to obtain a "quality profile" of the results of a single inspection or a group of inspections.

   Diagram 5-2 is a check sheet created from hypothetical data. The column of two-letter codes on the left represents all allowable Engineering Problem Type codes. Each X to the right of an Engineering Problem Type code represents a problem recorded at the inspection to which the code has been assigned. Thus, 2 problems classified as DD (Data Definition) problems were found, 15 problems classified as MX (Module interdependency) problems were found, no problems classified as DI (Data initialization) problems were found.

   You can easily construct this check sheet manually from the Inspection Problem Report Form. The data comes from the Engineering problem category column of the form. The number of tics in the "multiples" column is used to get the correct number of instances of each problem type. If there are 3 tics, for example, then the problem occurred 4 times, so you place 4 X's to

the right of the matching engineering code row in the check sheet.

Incidentally, check sheets of inspection data do not have to be drawn manually. A special software tool called IDAT has been developed to allow inspection teams easily to enter inspection data into a database and to create check sheets from the data. The tool is available from the Software Tools Clearinghouse.

As you look at Diagram 5-2, you will notice that no DI (data initialization) problems were recorded. What does this mean?

Does it mean that there were no DI problems in this module? If not, why not? Perhaps the total absence of DI errors is because of some excellent programming practices. If so, what are those practices, and how can they be applied to other aspects of the programming process?

On the other hand, if there were in fact some DI errors present in the module, but they were not found during the inspection, what does that tell us about the inspection process? Can the process be improved to find more of these types of errors? If the process cannot be improved, do we need other methods for finding these errors?

Many different types of check sheets are possible and potentially useful. Here is another suggestion for a check sheet:

A Check sheet of Quality Dimension Type problems can give you an indication of whether your project may be having special problems in an important quality area. For instance, if a major goal of your project is to improve maintainability, then you probably would want to know how many problems found by inspection were assigned the MT (Maintainability) code. A word of caution, however: the presence or absence of a large number of problems on a check sheet is an indicator of something unusual happening, but the check sheet by itself cannot tell you WHAT is happening. To find out what the absence or presence of X's means, read on, especially the section below entitled "Cause and Effect Diagram".

Another approach to using checksheets is to pool the data from inspections of several Authors of several documents. The team of authors could look for the existence of high or low frequency problem categories, and they could begin to ask whether there is something about the way they are running their project or their inspections that could be causing the check sheet to look the way it does.

Asking such questions is a very good way of beginning the process of change. Asking questions can lead to discovering problems, discovering the causes of the problems, and finally removing the causes of the problems.

DIAGRAM 5-2
==========

Check Sheet of Engineering Type Problems
Found at Inspection of Program "RXD.BLI"
(Hypothetical Example)

```
AM
DI
DD XX
DU XX
EP XX
GH
GO
IC X
LI XXXX
LO X
MX XXXXXXXXXXXXXXX
OO XX
PC XXX
PV
SY
UE
NS
TS
OT
```

2. Pareto Diagram.

You can easily draw a Pareto Diagram from a check sheet. The Pareto Diagram in Diagram 5-3 has been constructed from the data in the check sheet in Diagram 5-2.

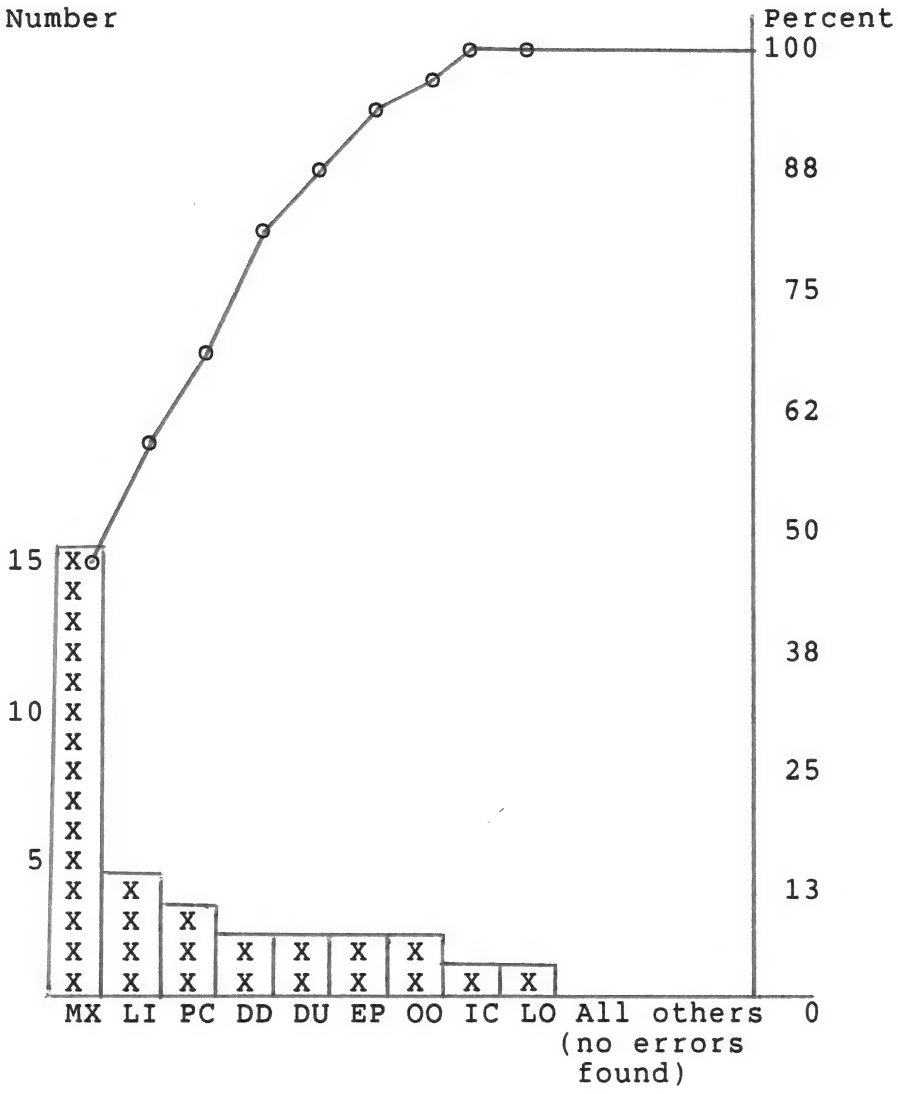
The example Pareto Diagram shows that about 50 percent of all problems were classified as MX problems, and that about 73 percent of all problems were classified as MX (module interdependency), LI (Loop initialization), and PC (Programming convention) problems. If we believed that this data was representative of most of the software produced on our project, and if we wanted to make a large change to the software development process in a hurry, we should first consider problems of type MX, LI, and PC, because they are the most numerous.

If we were not convinced that the data was representative of most of our project, we should collect more data, enough to give us confidence that we had an accurate "quality profile" of our project. We could achieve this confidence by inspecting everything, or by inspecting randomly selected modules, or by identifying key modules. A statistician's advice would be helpful in helping us to decide on a sampling scheme. We have some evidence that IBM practice is to inspect everything on a project, even though the cost seems very high. At DEC we have not investigated the costs and benefits thoroughly enough to be sure of the correct strategy.

In any case, the Pareto Diagram provides us with a rational indication of what aspect of our process would be easiest to improve.

DIAGRAM 5-3
==========

Pareto diagram of problems found
by inspection by engineering defect category
(Hypothetical Example)



3. Cause and Effect diagram.

Once we have decided on an area of our engineering process that we wish to improve, we need to discover the causes for problems in that area. This is one place in which Cause and Effect diagrams can be helpful.

The first step is to be sure that the situation, or effect, you want to explain is very clearly described. For example:

Effect: "An unexpectedly large number of MX problems were found at the inspection". Because this is a cumbersome phrase, just for the purposes of this description of Cause and Effect diagrams, let's label this statement: "Effect A".

The next step is to identify all possible causes for the effect, even the unlikely causes. A carefully produced Cause and Effect diagram provides an excellent way of representing these causes, and of showing which are the direct causes, and which are the subordinate causes.

Example 5-4, while simplified and constructed just for this explanation, illustrates how Cause and Effect diagrams work. Cause and Effect diagrams are best constructed by teams. Please imagine that Example 5-4 was constructed by a hypothetical team.

The team meets in a room that has a large black board (or white board). The author has seen conference rooms in Japan where the walls themselves are porcelain boards, so there is plenty of writing space for brainstorming. The team leader draws a horizontal line across the board, and facilitates team discussion to develop precise wording for the "effect" to be charted. The leader writes the effect to the right of the line.

The team then suggests as many possible causes for the effect as they can. No suggestion is turned down or even debated. The leader's main job is to encourage suggestions, and to structure them in the most meaningful way.

In the chart, the main, or direct, causes are represented by diagonal lines coming off the center line.

Thus, in Diagram 5-4, one of the main causes for "Effect A" was identified by our hypothetical group as the inspection process itself. We can read the line labelled "Inspection Process" as an effect:

"The Inspection Process caused Effect A". Again, for convenience in this discussion, let's call this Effect B. Effect B needs explanation in terms of its possible causes, also.

The two main causes identified for why the inspection process might be the cause for Effect A were identified by our hypothetical team as:

"Implementation (of the inspection process)" - Effect C,

and:

"Only good for finding MX problems" - Effect D.

We see three lines coming off the "Implementation" line. Looking at just one of them, "Review rate wrong", we can read it as:

"Review rate wrong" caused Effect C.
Effect C in turn caused Effect B.
Effect B caused Effect A.

Subsidiary causes for why its implementation was the cause of so many MX problems were identified as: poor code reading, poor planning, and wrong review rate.

A good Cause and Effect diagram might list 50 to 100 or even more possible causes for the effect cited. A good diagram will display a structure of causes and subcauses, several levels deep. These quotes from Dr. Ishikawa's book, "Guide to Quality Control", chapter 3, give us further insight into evaluating a Cause and Effect diagram:

"If a cause and effect diagram can be drawn thoroughly, it means those doing it know quite a bit about the production process."

"A cause generally consists of many complex elements. Therefore cause and effect diagrams usually turn out to be rather complicated"..."If one turns out looking like the one in figure 3.11 [many lines coming from the center line, but no other lines], it means that your knowledge of the... process is still too shallow. Also, if your diagram only consists of five or six causes, it cannot be considered a good diagram."

Once the chart has been completed, the team must decide which one, or which few, of the many causes listed are the most likely important causes. There will probably be several contributing factors to any cause, but often by changing just a few of these factors we can see a large improvement. The difficulty is knowing which factors to select.

The team should now try to establish a method for verifying whether they have chosen correctly. Sometimes a simple experiment can be devised to test the team's hypothesis. But if no other method is possible, a reasonable approach is to assume the selected causes are the right ones, and to make a few simple changes to the process. Then observe the results of the changes by continuing to create checksheets from the inspection data.

The ensuing data will show a change for the better, or no change, or a change for the worse. You may as well assume that your changes to the process were responsible for the difference, and accordingly decide to retain the changes or not.

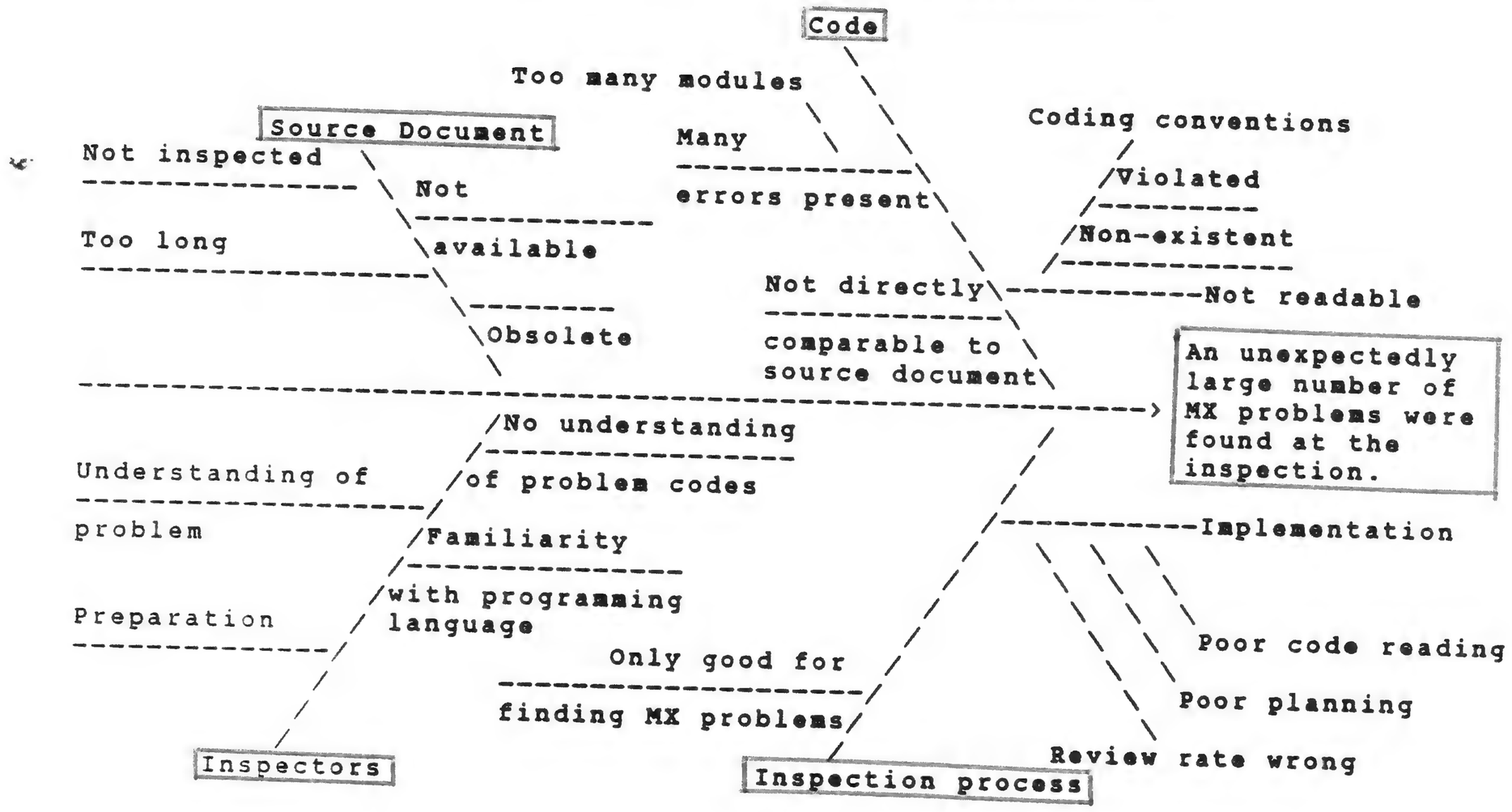
The process of producing check sheets, asking questions, and hypothesizing answers by means of Cause and Effect diagrams, should be done frequently during a project.

The benefits of using a process like this over many months are:

- Better team understanding of the engineering process
- Controlled and verified step-wise improvements to the engineering process
- Higher quality product because of improved process and improved inspections
- Greater productivity because of more problems found and fixed by inspections, instead of by costly testing

DIAGRAM 5-4

EXAMPLE OF CAUSE AND EFFECT DIAGRAM



4. U-chart. One of the common denominators between inspections is the rate of problem discovery per volume of inspected text, that is, the number of problems found divided by the number of lines of text inspected.

Diagram 5-6 is a u-chart that compares rate of problem discovery, expressed in problems per "Inspection Unit", for 10 hypothetical inspections. An "Inspection Unit" (IU), is a "standard" sized target document of size = 500 lines of text. The vertical axis represents the rate of problem discovery and the horizontal axis represents the inspections.

For example looking at Diagram 5-5, we see that inspection 4 was of a document having 450 lines, and that 23 problems were found. Thus the rate of problem discovery for inspection 4 was (23)/(450/500), or 25.6 problems per 500 lines of text.

Computation of a u-chart (based on hypothetical data): refer to Diagram 5-5:

Units for "Size of item inspected" must be decided at the outset, and should be chosen as a meaningful and convenient measure of size. For formal inspections, size has been defined as the number of non-commented-source-lines of code, or the number of lines of text, depending on the type of document inspected.

"Number of IUs" is the size of the inspected item expressed as a fraction of the standard size. For this example, a standard IU is 500 lines, so "Number of IUs" is "Size of item inspected" divided by 500.

"Number of problems found" is the problem count for an inspected item.

"Number of problems per IU" is: "Number of problems found" divided by "Number of IUs".

ubar is the average of the "Number of problems per IU" = average of the u's = 27.28

UCL and LCL must be computed for each inspection:

$$UCL = ubar + 3 * \frac{\text{square root (ubar)}}{\text{square root ("Number of IUs inspected")}}$$

$$LCL = ubar - 3 * \frac{\text{square root (ubar)}}{\text{square root ("Number of IUs inspected")}}$$

Interpretation of the u-chart:

If the rate of problem discovery for a particular inspection is not "substantially the same" as for a group of inspections done within the same project, we would want to try to identify the causes of the difference, and thereby take action to improve the process.

If the rate is "substantially the same" over many inspections, we would like to discover how to decrease it. We would want to identify those error categories with the highest frequencies and associated costs, and determine what actions should be taken to reduce the causes of such errors.

We would also want to identify error categories which have very low frequencies and try to determine whether the inspection process is unable to find these errors. If so, we know that we must change the process to make it more sensitive to these errors, or we must find other ways of finding the errors.

UCL and LCL stand for Upper Control Limit and Lower Control Limit, respectively. If the number of errors per IU for an inspection is either above the UCL or below the LCL, then we have statistical evidence that something happened at that inspection which was substantially different from the other inspections. This could be a warning that the target document is especially complex, or especially difficult to understand, or of especially low quality. Another possibility is that the inspection team was especially harsh or especially lenient on the author, for any number of reasons. In any case, the u-chart gives us the opportunity to ask questions about the engineering process. We can use check sheets, Pareto diagrams, or Cause and Effect diagrams to understand what happened.

In our example some points fall above and below the UCL and LCL (see inspections 1,2 and 7). But if we constructed a chart in which NO POINTS fall above the UCL or below the LCL, then we say we are operating within a system that is "under statistical control". The causes for errors found in any one inspection are probably substantially the same causes for the errors found in all the other inspections.

The important question to ask then is: are we content with the amount of variation from one inspection to the next? are we content with the average number of problems found per inspection? Being under statistical control means that we will continue to experience that much variation and that average number of errors forever, until we change the system that is producing the results.

Thus, u-charts give us a way of deciding if any particular inspection was different from the main body of inspections on our project, are were therefore subject to special causes for variation.

u-charts also tell us how much variation we can expect in problem discovery rate from one inspection to the next, if we are working within an unchanging system.

DIAGRAM 5-5

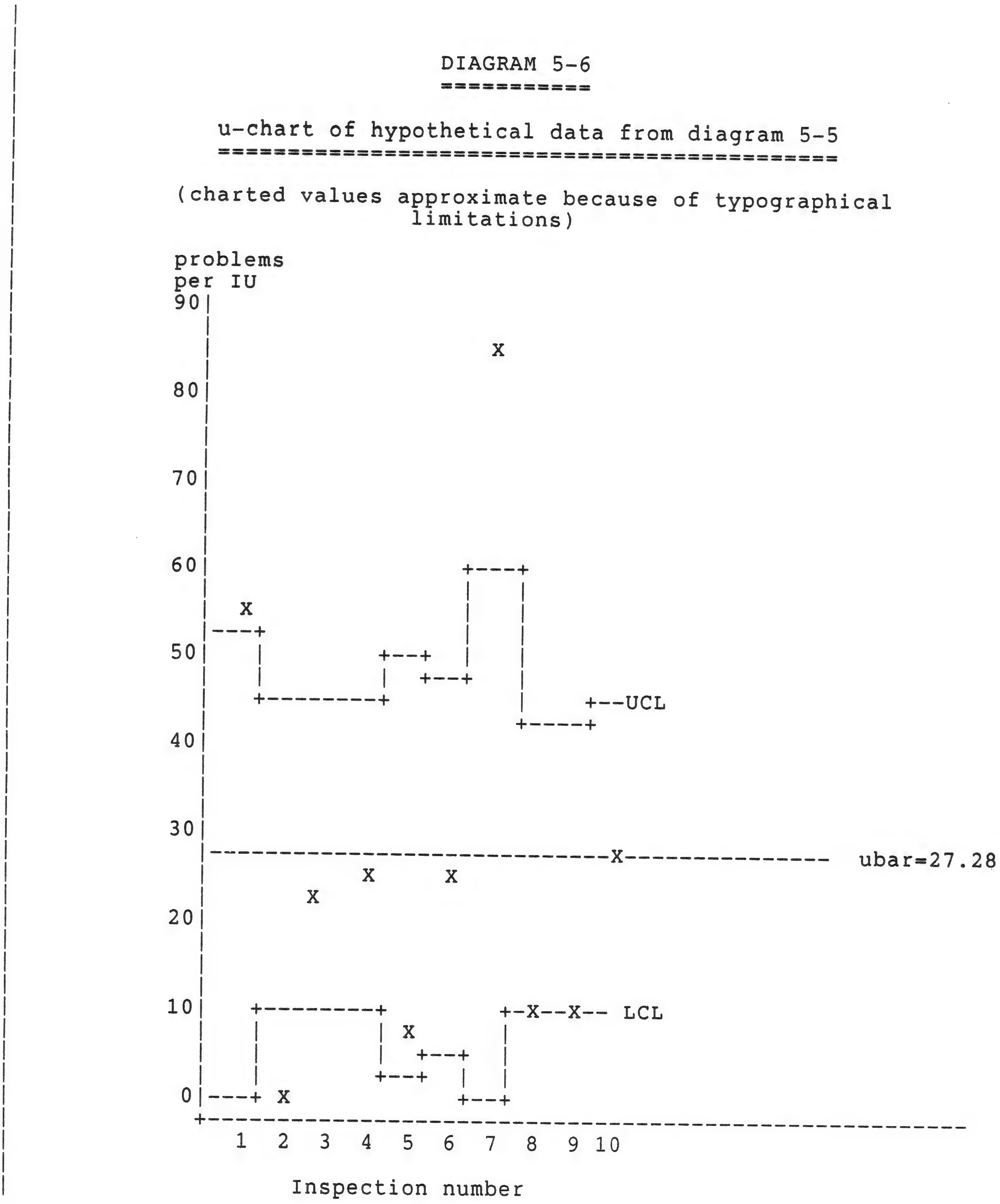
Computation of a u-chart

Inspection number	Size of item inspected	Number of IUs inspected (IU)	Number problems found	Number problems per IU (u)	UCL	LCL
1	181	0.36	20	55.2	53.4	1.2
2	375	0.75	4	5.3	45.4	9.2
3	483	0.97	22	22.8	43.2	11.4
4	450	0.90	23	25.6	43.8	10.8
5	211	0.42	3	7.1	51.5	3.1
6	284	0.57	14	24.6	48.0	6.5
7	123	0.25	21	85.4	58.6	-4.1(use 0)
8	452	0.90	9	9.9	43.8	10.8
9	488	0.98	9	9.2	43.1	11.5
10	415	0.83	23	27.7	44.5	10.1

DIAGRAM 5-6

u-chart of hypothetical data from diagram 5-5

(charted values approximate because of typographical limitations)



APPENDIX A

INSPECTION GUIDELINES - REQUIREMENTS

When inspecting a Requirements document it is convenient to divide the content of the document into "Outside" and "Inside" information.

"Outside" information relates to decisions expressed in the Requirements Document which have to do with marketing objectives. Usually such decisons are made intuitively, or at least by a variety of methods not familiar to the engineer. As far as the inspection process is concerned, these decisions should be regarded as correct (within reason). The inspection process has not yet been adapted to dealing with this decision-making process.

"Inside" information relates to the manner in which the Requirements have been expressed. The critical questions are: "Are the Requirements stated in such a way that an engineering organization can do business with them?" "Are they unambiguous, complete, and authoritative?" The inspection process CAN help to answer these questions, and the checklist which follows can be used as a guide to the inspector.

1. Is there a global statement of the intent of this product? Such a statement will not provide a measureable goal, but will provide a general direction. Example: "We want to capture the hearts of housewives".

2. Do requirements exist in each of the applicable software quality dimensions:

   - o Environment
   - o Features/Capabilities
   - o Publications
   - o Packaging
   - o Installability
   - o Human Factors

- o Performance
- o Reliability
- o Maintainability
- o Maintainence
- o Compatability
- o Evolvability
- o Cost
- o Timeliness
- o Constraints/Trades-off

3. Is each requirement unambiguous? (If the requirement were read aloud several times, with emphasis on the first word, then the second word, and so on, would the requirement take on different meanings?)

4. For each stated requirement, is there a way for any reasonable person to tell at the end of the project whether the product meets or does not meet the requirement? Has the way been stated in the requirements document?

5. Are there any unstated or missing requirements?

6. Are there any unnecessary requirements?

7. If there are any conflicting requirements, are there clear guidelines for how trades-off should be made?

8. Are TARGETS and CONSTRAINTS clearly labelled?

   * A TARGET requirement is a desirable but not absolutely necessary characteristic of the product.

   * Are the TARGETs given a clear priority ordering?

   * A CONSTRAINT is a requirement that must be present. Without fulfillment of the CONSTRAINT, the end product would be unacceptable.

APPENDIX B

INSPECTION GUIDELINES - DESIGN

1. Does the design help to meet at least one project goal?
2. Does the design implement any unnecessary functions?
3. Does the design identify all side effects and changed values?
4. Does the design properly address all human factors?
5. Is the design complete?
6. Is the design understandable? Unambiguous?
7. Is the design self-contained? Could someone brand-new to the project succesfully implement the project from the documentation alone?
8. Does the design show linkages between all applicable modules?
9. Does the design list all external dependencies?
10. Does the design take into account packaging and installation requirements?
11. Does the design list all applicable control blocks, tables, and data structures, and all new functions for which they are used?
12. Does the design identify all new macros, symbols, and coding conventions?
13. Does the design identify all specific values required and set?
14. Is the design written at the appropriate level of detail?
15. Is the design written at a consistent level of detail?

16. Does the design address all known possible cases?

17. Does the design address all exception cases?

18. Does the design address abnormal cases and error conditions?

19. Does the design address all appropriate operating environments/devices?

20. Does the design provide for new logic and function that is consistent with existing logic and function?

21. Are all fields described correctly?

22. Are there any missing fields?

23. Should an external routine be used rather than performing the function internally?

24. Does the module provide for all possible input parameters?

25. Does the module process all parameters correctly?

26. Does the module pass all parameters correctly?

27. Does the design provide for illegal input parameters?

28. Is all new/changed information flow across external linkages correctly described?

29. Are all return codes, parameter formats, and so on, correctly identified, or are there adequate references in this document to their definition?

30. Does the design provide for reentrance? reusability?

31. Are all issues of code protection addressed?

32. Have performance issues been addressed?

33. Has storage size been addressed?

34. Does the function use existing facilities wherever possible?

35. Are there any paging/swapping issues?

36. Have issues of maintainability been addressed?

37. Have issues of reliability been addressed?

38. Have issues of evolvability been addressed?

39. Have issues of functionality been addressed?

40. Have issues of compatability been addressed?

41. Have issues of user documentation been addressed?

42. Have issues of user training been addressed?

43. Have issues of software specialist training been addressed?

APPENDIX C

INSPECTION GUIDELINES - EASE OF USE

1. Minimize complexity in general.

2. Make the system appear simple to the naive user, yet extensible for the experienced user.

3. Reduce the number of different functions:

   - o Implement only those functions that will be needed frequently.
   - o Avoid providing numerous and complex "features".

4. Minimize keystrokes.

5. Make the functions provided "orthogonal".

6. Maximize the intuitiveness of the commands and displays:

   - o Do not invent strange new notations for old concepts.
   - o Maximize similarity to existing functions.

7. Reduce visual clutter.

8. Provide predictable response time.

9. Address the need for "closure". Allow user to successfully complete many small tasks rather than wait for one very large task.

10. Avoid pejorative error messages. Try to inform the user how to correct the problem.

APPENDIX D

INSPECTION GUIDELINES - GENERAL CODING

(From the Ethnotechnical Review Handbook,
by Weinberg and Freedman)

D.1 FUNCTION

1. Is there a concept, an underlying idea, that can be expressed easily in plain language? Is it expressed in plain language in the implemented code?
2. Does the function of this part have a clear place in the overall function of the whole? Is this function clearly expressed?
3. Is the routine properly sheltered, so that it can perform its function reliably in spite of possible misuse?

D.2 FORM

1. Is the style clean and clear?
2. Is it meaningful to all classes of readers who will see it?
3. Are there repeated code segments, whether within or between routines?
4. Are comments useful or are they simply alibis for poor coding?
5. Is the level of detail consistent?

6. Are standard practices used?

7. Is initialization properly done, and does the routine clean up after itself?

D.3 ECONOMY

1. Are there redundant operations for which there is no compensating benefit?

2. Is storage use consistent, both internally and with external specifications?

3. How much will it cost to modify? (Consider the three most likely modifications.)

4. Is it simple?

APPENDIX E

INSPECTION GUIDELINES - BLISS

E.1 COMPILE PROBLEMS

1. Semicolon use.
2. Discarding a return value.
3. Cutting off an ELSE.
4. Macro substitution.
5. Quote and unquote levels.

E.2 RUN TIME PROBLEMS

1. Logic problem.
2. Loop count off; loop limit problem.
3. Variable not initialized.
4. Locals not initialized. OWNs and GLOBALs get initialized to zero, but one should not count on this.
5. Missing dot.
6. Too many dots.
7. Are dots used correctly for variables versus literals?
8. Are dots used correctly on parameters being passed to another routine?
9. If a passed parameter is REFed or not (more on dots).

10. Are there transportability problems such as using 16 of 32 bits, or mixing string and vector versions of character data?

11. The CH$ functions can be a source of trouble. Some change the character pointer as they are writing or reading characters and use the actual pointer address, whereas some do not require the value of the pointer. ("I don't know how many times I've changed a CH$RCHARA (ptr) into a CH$RCHAR (ptr) and forgotten to also dot the ptr").

12. Return values on routines: are they pointers, values, or what?

13. People writing -10/20 code often confuse the BLISS "AND" with the bit test operation performed by the machine Logical Testing and Modification instructions. The typical erroneous code sequence is:

```
IF .AC AND MASK THEN ...
```

    It should be:

```
IF (.AC AND MASK) NEQ 0 THEN ...
```

14. Are begin-ends balanced?

15. Zero-or-one problem: Do you address the first element of a data structure as x[0] or x[1]?

16. Are maps and binds used correctly?

17. In code that defines data structures, do the structures look as though they belong together?

18. Are STRUCTURE definitions correct?

19. External NOVALUE routine used in a value context.

20. Typos: symbol names mistyped, missing or extra commas and semicolons.

21. If you have a reference to a field of a structure such as:

```
(IF expr THEN A<0,8,5> ELSE B<0,8,5>) = .C
```

    you expect that the value at C will be stored in a restricted field, either A<0,8,5> or B<0,8,5>. Instead, the size of the field is of no interest and the value in C gets stored into A<0,8,32> or B<0,8,32> (in VMS).

22. Macros defined in the far reaches of common libraries: are they being used correctly?

APPENDIX F

GUIDELINES FOR TEST PLANS

1. Does the Test Plan describe clearly and completely what is to be tested? ...In terms of (for example):

    1. Function?

    2. Environment?

    3. Quality Dimensions?

    Does the Test Plan describe an adequate set of Test Cases, covering both "normal" and "abnormal" situations which the product must deal with?

2. Is the testing approach feasible?

3. Are all the new and changed User Interfaces covered by the Plan?

4. Are the defaults adequately covered by the Plan?

5. Does the plan cover all the error conditions, in terms of...

    1. how the Product reacts in the case of errors, and in terms of

    2. the appearance and value of the error messages?

6. Are regression tests included in the plan?

7. Are there any controversies still active which would change the value of the Plan?

APPENDIX G

GUIDELINES FOR TEST CASES

1. Has every Test Case described in the Test Plan been implemented?

2. Is each Test Case completely and accurately identified and described?

3. Are there clear and complete operating instructions for each Test Case?

4. Is each Test Case as self-checking as is possible?

5. If appropriate, does the Test Case issue messages which notify the user that the test is in progress?

6. Does the Test Case comply with all the requirements of the Test Plan?

APPENDIX H

GUIDELINES FOR INSPECTION RATES

	Preparation Rate	Inspection Rate
Most text documents, such as Requirements, Specs, Plans, and High-Level Design	250 - 500 lines/hr	250 lines/hr
Code (any language) or other machine - processable material	125 - 250 NCSS/hr	125 NCSS/hr

NOTES:

* NCSS = "noncommented source statements". That is, number of lines of code after lines of comments have been subtracted out. For users of BLISS and other languages in which "lines" can be defined by syntax rather than by appearance on the printed page, please define a "line" as one printed line (excluding comments).

  Code inspections at DEC so far have been at far faster rates than recommended here. These same inspections have also discovered fewer important problems in the code than are found by inspections of text documents.

APPENDIX I

SAMPLE PROBLEM REPORT FORM

INSPECTION PROBLEM REPORT FORM

Page ____ of ____

Date of Inspection:_____

Target Document:_____

Recorder:_____

Please send a copy of Inspection Report and the Inspection Problem Report to Lou Cohen, CQG, ZKO 1-2/D13

Moderator:_____

Problem ID	Problem Description	VI/ IV EX	MS/ WR/	Qual	Qual	Eng	Multiples	Disagreer

Revised 2 August 1983

Problem ID	Problem Description	VI/ IV	MS/ WR/ EX	Qual	Qual	Eng	Multiples	Discoverer

APPENDIX J

PROBLEM CATEGORY CODES AND OTHER CODES

TYPE ONE - VISIBILITY

VI Visible to user
IV Invisible to user

TYPE TWO - PRESENCE

MS Missing
WR Wrong
EX Extra

TYPE THREE - QUALITY

CO Costs
CM Compatibility

CN Constraints and trade-offs
EN Environment
EV Evolvability
FE Features; functionality
HF Human factors, ease of use
IN Installability
MN Maintenance (bug fixing)
MT Maintainability
PB User publications
PE Performance
PK Packaging
RE Reliability
TM Timeliness
HS Not classified

TYPE FOUR - ENGINEERING

AM Statement ambiguous
DI Data initialization
DD Data definition or declaration
DU Data usage
EP Explanation or definition
GH Goal in previous document found to be a problem during this inspection
GO Contradicts goal of project
IC Inconsistency between statements in this document
LI Loop initialization error
LO Logic
MX Module interdependency
OO Count off by one
PC Programming convention
PV Passive voice
SY Syntax
UE Use of English
NS Not specific enough for this document
TS Too specific for this document
OT Other

Codes for type of material inspected:

01 Requirements	04 Detailed design	07 Documentation plan	10 Test cases
02 Specification	05 Development plan	08 User Documentation	11 Help files
03 High-level design	06 Test plan	09 Code	12 _____

Codes for reasons why inspection did not occur:

1 One or more inspectors not available	5 Inspection cancelled by management
2 One or more inspectors not prepared	6 Conference room not available
3 Material not ready in time	7 Other
4 Material judged to be not inspectable	

Revised 2 August 1983

APPENDIX K

SAMPLE INSPECTION REPORT FORM

digital

I N S P E C T I O N R E P O R T

MK01-1/AG5

Please send a copy to Lou Cohen, ~~ZKO 1-2 / D13~~
Note: fields in boxes are for input to IDAT

To: _____ From: _____ (Badge)
(Supervisor/ Project Leader) (Moderator)

Project Leader: _____ Date of inspection: M M D D Y Y

Product/
Project/
Sub-project
(select an alphanumeric ID which suits your data collection requirements)

Type of material inspected: (See reverse side for codes) 9 9

Number of lines of material inspected 9 9 9 9

Inspection occurred as scheduled: Y/N

If inspection did not occur as scheduled, give reason: 9
(See reverse side for codes)

Reinspection required: Y/N

Is this a reinspection? Y/N

If this is a reinspection, enter the number of rework hours since previous inspection: 9 9 9 9

INSPECTORS	Prep hrs	Insp hrs	Phone	Netmail Address
Author:				
Moderator:				
Reader:				
Recorder:				
Number of people: 9 9	Total hours: 9 9.9 9	9 9.9 9		

(Revised 2 August 1983)

APPENDIX L

GLOSSARY OF TERMS

Author.
The person who wrote the target document.

Cause and Effect diagram.
One of the seven basic tools. It provides a team of engineers with a method for determining all the possible causes for some observed or hypothetical effect.

Categories.
See Problem Classification.

Check sheet.
One of the seven basic tools. One type of check sheet provides a simple way of classifying and recording events. Another type of check sheet is a memory aid.

Classification.
See Problem Classification.

Common causes of problems.
A set of causes for problems in a process that affects all people, machines, materials or methods in the process.

Control chart.
One of the seven basic tools. It provides a method of comparing one measured value with a set of others to determine whether the new value is due to the same set of causes which produced the previous values.

Fishbone chart.
See Cause and Effect diagram.

Formal process.
A process which has a predetermined outward form. The formal inspection process has an elaborate predetermined outward form. The purpose of such an elaborate set of rules is to encapsulate and communicate the best technical review practices to make them available to the largest audience.

Histogram.
One of the seven basic problem solving tools. It provides a graphic representation of a frequency distribution.

Ishikawa diagram.
See Cause and Effect diagram.

Inspection.
The process of examining a document for correctness by comparing it with the document(s) from which it was derived.

Inspection phase.
One of the six phases of every inspection. The inspection phase is initiated by the moderator. All inspectors are present. The reader reads the document aloud, and each inspector calls out and classifies problems as they are seen. Problems are recorded by the recorder. Duration of this phase depends on the size of the material being inspected. No inspection should last more than 2 hours.

Inspector.
One of several roles taken on by participants of a formal inspection. Every participant assumes the role of inspector in addition to any other roles assumed. The inspector's job is to spend one to two hours preparing for each two hour inspection, finding and recording problems as described in this handbook. The inspector is also required to arrive at the inspection on time, and with all necessary materials in hand. The inspector is also required to find as many problems with the target document as possible, during the inspection.

Line of code.
For purposes of the formal inspection process, a line of code is the information in a listing that fits on a single horizontal line.

Line of text.
The information printed on a horizontal line of a document. This definition is based on how people see lines of text. Not to be confused with various technical definitions of "line" that relate to how data is represented and processed by computers.

Moderator.
One of several defined roles taken on by those who participate in a formal inspection. There is one moderator involved with each inspection. The moderator is primarily responsible for the proper execution of the formal, or external aspects of the inspection. The external aspects include are the subject of this handbook. The moderator should also try to function as an inspector, if time and energy permit.

NCSS, Non-commented-source-statements.
Refers to lines of text in a software listing which are not entirely comments. The number of NCSS in a software listing is used in the inspection process as a rough measure of how much

work the listing represents.

Overview phase.
One of the six phases of every inspection. The Overview is initiated by the moderator. The answers to the basic what?, who?, when?, where? questions, answered during planning, are communicated to all the inspection participants by the moderator. Duration of this phase is 15 minutes.

Pareto Diagram.
One of the seven basic tools. It provides a graphical method for comparing the relative frequency of several measured factors.

Peer.
A fellow worker at the same level.

Phases of an inspection.
Each inspection consists of 6 phases: Planning phase, Overview phase, Preparation phase, Inspection phase, Report Results phase, and Rework phase. The phases are executed in sequence. Entrance criteria and responsibilities for each phase are described in this handbook.

Planning phase.
One of the six phases of every inspection. Planning is initiated by the author, and is then the responsibility of the supervisor. The basic what?, who?, when?, where? questions are answered during planning. Duration of this phase is 15 to 20 minutes.

Preparation phase.
One of the six phases of every inspection. Each inspector prepares for the inspection individually by studying the target document. The goal is find, record, and classify as many problems in the target as possible before the inspection begins. Duration of this phase is 1 to 2 hour for every 2 hours of inspection.

Presence code.
See Problem Classification.

Problem.
An inconsistency in the target document, as compared to the source document.

Problem Classification.
When each problem is recorded, the person who found the problem assigns certain codes to it to indicate what kind of problem it is. The codes indicate: whether the problem, if left unfixed, will be visible to users; whether the problem is because of a problem with existing material, or because of extra or omitted material; which quality dimensions are affected by the problem; and what kind of engineering error the problem is. These four different types of the problem description are called: Visibility type, Presence type, Quality Dimension type, and Engineering type.

Process.
A series of actions or operations contributing to some result.

Process control.
An activity which seeks to improve the effectiveness of a process.

Quality control.
An activity which seeks to improve the quality of a product.

Quality Dimension.
See also Problem Classification. The Quality Dimensions are the subject of the Quality Workshop and deserve considerable attention. The basic idea is that product quality is judged by the user of the product. Therefore, in order to understand product quality, we divide the user's perceptions into a set of manageable "Quality Dimensions". We then specify, build, and measure quality by treating each dimension separately (even though there is considerable overlap between the dimensions). The quality dimensions are: Cost, Compatibility, Environment, Evolvability, Capabilities, Human Factors, Installability, Maintenance, Maintainability, Performance, Packaging, Publications, Reliability, and Timeliness. One additional dimension not seen by the user has been added to the list: Constraints and Trade-offs.

Reader.
One of several roles taken on by participants of a formal inspection. There is only one reader at each inspection. The reader's job at the inspection is read the document aloud to the other inspectors. For technical documents this is a particularly difficult job, since the reader must understand the document well enough to provide an interpretation for everyone else.

Recorder.
One of several roles taken on by participants of a formal inspection. There is only one recorder at each inspection. The recorder's job is to record the problems found during the inspection. The method for recording is described in this handbook.

Report Results phase.
One of the six phases of every inspection. This phase is initiated by the moderator. The moderator and the complete the standard inspection report forms and distribute them the supervisor, the author, and the project statistician. Duration of this phase is 5 minutes.

Rework phase.
One of the six phases of every inspection. Initiated by the author. The author resolves all problems found at the inspection to his or her satisfaction. Duration of this phase depends on the problems found.

Scatter diagram.
One of the seven basic tools. It provides a simple graphical method for determining the relationship between two measurable variables in a process.

Sizing.
The process of counting the number of text lines on each page of a target document, and using the count to estimate how long it will take to inspect the entire document as well as each of its pages.

Source document.
One of possibly a few documents which form the basis for the contents of the target document.

Special causes of problems.
One or a few causes of problems in a process which affects a minority of the people, machines, materials or methods in the process.

Supporting document.
A special type of source document, which has a less direct influence on the target document than does the source document. For instance, suppose the target document is the development plan for a COBOL compiler project. The source document may be a Product Specification stating among other things the quality goals of the project, while the supporting document may be the ANSI specification for COBOL.

Target document.
The document to be inspected. The target is considered to have been derived from one or more source documents. The target is usually an elaboration and refinement of at least some of the aspects of the source document(s). The person who created the target document is called the author.

Teamwork Tools.

Another name for the Seven Basic Tools. The author prefers this name because it emphasizes the vital role these tools can play in fostering efficient team problem solving.

U-chart.
A special form of control chart especially well suited to comparing the number of errors found from an inspection with the numbers of errors found in a set of previous inspections. The U-chart provides statistical evidence for determining whether the causes for errors found in the new inspection are the same as the causes for the errors found in the previous inspection.

APPENDIX M

BIBLIOGRAPHY

Annotations are by Lou Cohen.

1. Gerald M. Weinberg and Daniel P. Freedman, "Handbook of walkthroughs, inspections, and technical reviews: evaluating programs, projects, and products", Little Brown, 1982, 3rd Edition. Two previous editions were published by the Ethnotech Press. The had a different title at that time: "The Ethnotechnical Review Handbook". The new edition is somewhat revised, and more slickly produced.

   The book is a common sense exposition of why formal reviews are important. It transcends any one review process and gets at the fundamentals. As with other Weinberg books, it is witty, easy to read, and convincing.

2. F. O. Buck, "Indicators of Quality Inspections", IBM TR21.802. This technical report defines and measures some quality factors of inspections. It provides the statistical basis for concluding that the most critical factor affecting the effectiveness of a code inspections is the rate of inspection speed. The next most critical factor is preparation time.

3. M. E. Fagan, "Design and Code Inspections to Reduce Errors in Program Development", IBM Systems Journal, vol 15, no. 3, 1976 (reprint available from IBM as order number G321-5033). This is an update of a previous article by Fagan which introduced formal inspections for the first time. Fagan was the originator of formal inspections. Unfortunately, these articles do not describe the process itself in much detail, only the results of some experiments. While the results are interesting, they make heavy use of IBM software development terminology. Therefore they may not strike a responsive chord to the DEC reader.

4. "Inspections in Application Development - Introduction and Implementation Guidelines", IBM user document GC20-2000-0. Similar in scope and content to the DEC "Inspector's Handbook", but it reflects the IBM style of formal inspections. The DEC style started with the IBM style and

evolved to be compatable with the DEC engineering culture.

5. Several handbooks on inspection methodology as practiced at IBM:

   - O. Robert Kohli, "High Level Design Inspection Specification", IBM TR21.601.
   - O. Robert Kohli and Ronald A. Radice, "Low Level Design Inspection Specification", IBM TR21.629.
   - Rodney R. Larson, "Test Plan and Test Case Inspection Specification", IBM TR 21.586.
   - Joseph Ascoly, Michael J. Cafferty, Stephen J. Gruen, and O. Robert Kohli, "Code Inspection Specification". IBM TR 21.630

6. W. Edwards Deming, "Quality, Productivity, and Competitive Position", MIT Center for Advanced Engineering Study, 1982. The best book available on the management philosophy of quality. It provides the bridge between the statistical and the philosophical implications of control charts.

7. Dr. Kaoru Ishikawa, "Guide to Quality Control",Asian Productivity Organization, Tokyo, (eleventh printing, August 1983). Can be obtained through UNIPUB, Box 433, Murray Hill Station, New York, NY, 10157, tel: 800-521-8110. Provides excellent summaries of how to use each of the Seven Basic Tools, plus a few more tools. Unfortunately, this book, as is true for all other books on quality control, takes all of its examples from the production line. Many engineers have difficulty relating the material to their own jobs, but it must be done, and we don't have much more to work with, yet. Still, it is one of the best books of its type in English.

8. Eugene L. Grant and Richard S. Leavenworth, "Statistical Quality Control", McGraw Hill, 1980 (5th edition). A standard U.S. text book on statistical quality control, although experts find fault with sections here and there.